Universidad Carlos III de Madrid
Escuela Politécnica Superior

DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA EXPERTO DIAGNOGSYS

Proyecto Fin de Carrera

Ingeniería Técnica
en Informática de Gestión

Autor: Alejandro Halcón Fernández
Tutor: Ángel García Crespo

A las mujeres de mi vida:

a mi abuela paterna, María Josefa,
a mi madre, María Teresa,
a mi mujer, Sonia
y a mi hija, Emma.

AGRADECIMIENTOS

Quiero agradecer el apoyo prestado por mis padres, María Teresa y Julio, sin el cual no habría podido cursar mis estudios universitarios y no habría podido llevar adelante este proyecto.

También quiero agradecer muy especialmente el apoyo incondicional y comprensión que me ha prestado siempre mi mujer, Sonia, en todo este tiempo.

Por último, quiero agradecer a mi tutor, Ángel García Crespo, que me haya permitido realizar este proyecto bajo su dirección y la paciencia infinita que ha demostrado tener.

Índice

1. INTRODUCCIÓN

1.1. Motivación y objetivos

El presente proyecto consiste en el diseño y la implementación de un sistema experto multidisciplinar y polivalente capaz de resolver problemas que se le plantearían a un experto humano de una determinada materia o disciplina.

Este documento explicará y detallará el diseño y el funcionamiento del sistema experto Diagnogsys aplicado al ámbito de los diagnósticos médicos, cuyo principal objetivo consistirá en simular a un experto humano en medicina general, como ejemplo del funcionamiento de Diagnogsys, aunque, como ya se ha explicado, podría aplicarse a cualquier otra materia. Bastaría con proporcionar al sistema el escenario inicial propio de la disciplina en la cual se quiera que el sistema sea un sistema experto.

Para conseguir crear un sistema basado en los sistemas expertos se utiliza el concepto de la Inteligencia Artificial, que consiste en el estudio de la realidad expresándola de tal manera que pueda ser manipulada por una máquina inteligente.

Un sistema experto se define como un sistema que simula a los expertos humanos en un área determinada. Este sistema pretende simular a un experto en medicina general, lo que llamamos comúnmente un médico de cabecera, que diagnostica la enfermedad que tenemos gracias a unos síntomas que el paciente padece y le comunica a su médico.

El **objetivo** de este proyecto es desarrollar un sistema experto que sea capaz de proporcionar un diagnóstico médico, a partir de unos parámetros de configuración, y de los síntomas que el paciente ha experimentado.

El sistema pretende ser sencillo, por lo que, debido al amplio campo de la medicina, los parámetros de configuración que van a definir los síntomas que el paciente puede manifestar y los diagnósticos posibles se van a limitar a casos muy comunes en las consultas médicas como pueden ser resfriados, gastroenteritis o cefaleas, por poner algunos ejemplos.

1.2. Estructura de la memoria

La memoria de este proyecto está estructurada en 8 capítulos.

CAPÍTULO 1. Se ofrece una introducción al tema del proyecto.

CAPÍTULO 2. Se explican los fundamentos teóricos del proyecto: un conjunto de definiciones y conceptos médicos necesarios para el desarrollo y entendimiento del sistema, una introducción a la inteligencia artificial, un breve recorrido por los distintos campos que engloba esta materia y una explicación más detallada de los sistemas expertos centrándose en aspectos

como las formas de representar el conocimiento, el tratamiento de la incertidumbre, etc.

CAPÍTULO 3. Trata las distintas herramientas que se pueden encontrar en el mercado y los métodos apropiados para la elección de un entorno para construir sistemas expertos.

CAPÍTULO 4. Se establece una descripción de los fundamentos teóricos de la adquisición del conocimiento y de los diferentes métodos que hay para extraer el conocimiento de un experto.

CAPÍTULO 5. Se habla del desarrollo del sistema experto.

CAPÍTULO 6. Presenta un manual de usuario para el manejo del sistema.

CAPÍTULO 7. Se muestran las conclusiones que se derivan de la realización del presente trabajo, evaluando los resultados obtenidos y proponiendo nuevas ideas para futuros desarrollos.

CAPÍTULO 8. Se incluyen las referencias bibliográficas.

2. INTELIGENCIA ARTIFICIAL Y SISTEMAS EXPERTOS

2.1. Conceptos Médicos

En medicina, el **diagnóstico** o propedéutica clínica es el procedimiento por el cual se identifica una enfermedad, entidad nosológica, síndrome, o cualquier condición de salud-enfermedad (el "estado de salud" también se diagnostica).

En términos de la práctica médica, el diagnóstico es un juicio clínico sobre el estado psicofísico de una persona; representa una **manifestación** en respuesta a una demanda para determinar tal estado.

Diagnosticar es dar nombre al sufrimiento del paciente; es dar una "etiqueta".

El diagnóstico clínico requiere tener en cuenta los dos aspectos de la lógica, es decir, el análisis y la síntesis, utilizando diversas herramientas como la anamnesis, la historia clínica, exploración física y exploraciones complementarias.

El diagnóstico médico establece a partir de síntomas, signos y los hallazgos de exploraciones complementarias, qué enfermedad padece una persona. Generalmente una enfermedad no está relacionada de una forma biunívoca con un síntoma, es decir, un síntoma no es exclusivo de una enfermedad. Cada síntoma o hallazgo en una exploración presenta una probabilidad de aparición en cada enfermedad.

2.2. Introducción a la Inteligencia Artificial

Muchas actividades del hombre tales como escribir programas para ordenadores, hacer deducciones matemáticas, razonar en base al sentido común, entender un lenguaje e incluso conducir un automóvil, se dice que requieren inteligencia.

A lo largo de las últimas décadas se han construido sistemas informáticos y ordenadores capaces de realizar tales tareas. Por ello, podríamos decir que esos sistemas poseen cierto grado de inteligencia artificial.

El término "inteligencia artificial" se debe a Jhon McCarthy, que lo acuña a mediados de los años 50 al organizar un congreso sobre nuevas perspectivas de la investigación en informática. El congreso se celebró en Dartmouth (E.E.U.U) en 1956. En él aparecieron realizaciones de sistemas con capacidad para desarrollar juegos (juego de damas de Samuels), demostrar teoremas (The Logic Theorist de Newell y Simón), etc. Esta es la fecha que se suele considerar como la del nacimiento oficial de la disciplina.

La línea de trabajo que se identifica con la etiqueta de inteligencia artificial ha desarrollado y organizado un conjunto de ideas, técnicas y productos que constituyen todo un nuevo ámbito disciplinar.

Pablo Adarraga y José Luis Zaccagnini en su libro *Psicología e Inteligencia Artificial* ponen de manifiesto que la inteligencia artificial no es más que una determinada forma de programar ordenadores, es decir, de utilizarlos como herramientas para realizar tareas y resolver problemas, que se distingue de la informática convencional tanto por el tipo de problemas que aborda como por la forma en que lo hace.

¿A qué se debe el nacimiento de la inteligencia artificial? En pocas palabras, a la imposibilidad de resolver problemas, como una partida de ajedrez o un diagnóstico médico, donde no es posible describir exactamente y exhaustivamente todos los pasos a realizar para alcanzar el resultado deseado, mediante programación convencional.

La inteligencia artificial surge para abordar problemas y tareas no algorítmicamente tratables mediante el diseño de programas de ordenador que implementan procedimientos no exhaustivos (heurísticos), inspirados en el funcionamiento de la mente humana, que es capaz de realizar tareas como las antes descritas con un grado razonable de éxito, utilizando no sólo cálculos numéricos sino diferentes tipos de estrategias.

2.3. Áreas de la Inteligencia Artificial

Los métodos y las técnicas de la inteligencia artificial se han aplicado a diversos tipos de problemas. A continuación se describen algunas de estas aplicaciones.

2.3.1. Razonamiento Simbólico

Trata de emular la forma de razonar que tiene la mente humana cuando se enfrenta con tareas que no son resolubles mediante cálculo.

Lo que se hizo fue descomponer el conocimiento de los expertos en reglas del tipo:

SI: condición

ENTONCES: realícese determinada acción

Mediante éste mecanismo llamado "producción", es posible describir con bastante exactitud, todo el conocimiento que utiliza, por ejemplo, un jugador de ajedrez experto a la hora de tomar decisiones sobre la siguiente jugada.

Para emular el razonamiento simbólico heurístico (no algorítmico, no exhaustivo), que utiliza la mente humana para resolver problemas que no son abordables mediante cálculo puro, es necesario dotar a la máquina de dos cosas:

• Una representación simbólica para representar el conocimiento del medio sobre el que ha de operar.

• Algún tipo de estrategia inferencial capaz de operar sobre dicha representación simbólica, para extraer conclusiones a partir de datos sobre el estado en que se encuentra el medio, en un momento dado.

Este último mecanismo busca un camino a través de la representación simbólica para alcanzar la solución o soluciones posibles.

2.3.1.1. Sistemas Expertos

Un sistema experto es un sistema informático que incorpora, en forma operativa, el conocimiento de una persona experimentada, de forma que es capaz de responder como esta persona, de explicar y justificar sus respuestas.

El problema clave de los sistemas expertos es encontrar la forma de representar y usar el conocimiento que los humanos expertos poseen, ya que en muchos casos este conocimiento es impreciso, dudoso o anecdótico.

2.3.1.2. Razonamiento Subsimbólico

Si se quiere que un ordenador funcione como una mente humana, ¿por qué no se programa para que funcione como un conjunto de neuronas conectadas como las del cerebro humano?. Esta pregunta ha dado lugar a lo que hoy conocemos como redes neuronales.

Las redes neuronales no operan en el plano simbólico, funcionan a modo de filtros capaces de aprender a identificar determinados tipos de patrones numéricos, a partir de una cantidad suficiente de ejemplos.

Se asemeja bastante a cómo, por ejemplo, se aprenden las letras del abecedario, no basándose en una serie de reglas, sino aprendiendo a diferenciar unas letras de otras por puro ejercicio.

Las redes neuronales son una formalización matemática que se aproxima a un tipo de aprendizaje discriminal.

2.3.2. Tratamiento del lenguaje natural

Este área de la inteligencia artificial tiene su origen en los trabajos de Chomsky y en los desarrollos de la psicolingüística y pone de manifiesto que para que una persona sea capaz de comprender y producir disensos lingüísticos es imprescindible que se posea algún tipo de representación del escenario al que se refiere el discurso.

Parece por tanto, que un sistema informático capaz de comprender un mensaje en un lenguaje natural requerirá, tanto un conocimiento del contexto como un proceso para realizar las inferencias desde ese conocimiento.

Dentro de este campo es importante destacar la colaboración entre la inteligencia artificial, la lingüística y la psicología del lenguaje.

2.3.3. Visión artificial

Bajo este nombre se estudian las estrategias inteligentes que permiten a un ente interpretar las imágenes que capta del medio.

Una escena visual, por ejemplo, es codificada por medio de sensores y representada como una matriz de valores de intensidad. Estos valores son procesados después por detectores que los exploran buscando componentes primitivos de figuras. El objetivo final es representar la escena mediante un modelo adecuado. >: • ¿ :

2.3.4. Robótica

La robótica es el estudio de los mecanismos de control que permiten a un ente mecánico moverse en un medio físico y/o manipular elementos físicos con cierto grado de autonomía.

La investigación sobre robots ha ayudado al desarrollo de muchas ideas de inteligencia artificial.

Este área ha conducido a varias técnicas para modelizar estados del mundo y para describir los procesos de cambio de uno de estos estados a otro.

Además ha conducido también a una mejor comprensión de cómo generar planes para secuencias de acciones y cómo dirigir la ejecución de esos planes.

Muchos autores no consideran a la robótica como parte de la inteligencia artificial, sino a ésta como ingrediente de aquélla.

2.4. Ideas básicas para resolver problemas de Inteligencia Artificial

A continuación se indican, de forma esquemática, varias ideas básicas de cómo se pueden resolver problemas de inteligencia artificial.

1. El conocimiento se compone de: reglas, pareceres y heurísticas (descubrimiento de hechos):

   conocimiento = reglas + pareceres + heurísticas

2. Para conseguir el éxito del proyecto de inteligencia artificial se ha de buscar una solución necesaria y suficiente con los recursos válidos.

3. Búsqueda directa de las causas que puede producir ese éxito.

4. Como ayudas en la búsqueda de las causas se tienen:

- El conocimiento debe de ser aplicable, correcto y no necesariamente discriminatorio.
- Rápida eliminación de "callejones sin salida".
- Eliminación de cálculos redundantes en la computación.
- Aumento de la velocidad de las operaciones del ordenador.
- Búsqueda de caminos múltiples y cooperativos de conocimiento.
- Razonamiento en varios niveles de abstracción.

5. Eliminación de caminos que aumentan la dificultad del problema. Entre los que cabe destacar:

- Errores en el conocimiento o en los datos.
- El número de posibilidades a evaluar.
- Complejos procedimientos de desechar posibilidades.

2.5. Sistema Experto

Un sistema experto es un sistema software que trata de reproducir el comportamiento de uno o más humanos “expertos” ante un problema específico. Hay una amplia variedad de maneras para simular dicho comportamiento de los expertos, sin embargo, para que el sistema experto sea útil ha de tener las siguientes dos capacidades:

- Base del conocimiento: viene representada por un conjunto de reglas o pasos comprensibles de manera que se pueda generar la explicación para cada una de estas reglas, que a su vez se basan en hechos.

- Motor de inferencia: es un conjunto de mecanismos de razonamiento que permiten modificar los conocimientos anteriormente mencionados.

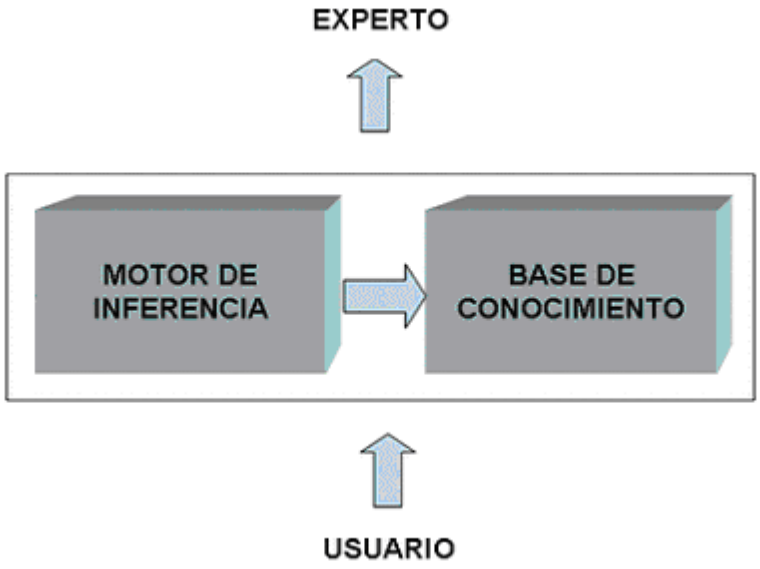


Figura 2.1: Arquitectura de un sistema experto

Una definición que aúna ambos conceptos se presenta a continuación.

Sistema Experto (DION, 2006) - sistema de información que representa el conocimiento de un experto dentro de un área particular de un problema como un conjunto de reglas, y lleva a cabo procesos de inferencia cuando se introducen nuevos datos.

Entre las aplicaciones originarias de la Inteligencia Artificial y la computación en cuanto a sistemas expertos se refiere, se encuentran la teoría de sistemas, operaciones de investigación, matemáticas aplicadas, etc.

Como ejemplo de aplicación de los sistemas expertos se puede mencionar el caso real de una refinería química en la que un empleado importante estaba a punto de retirarse, y la empresa veía en ello un grave problema puesto que era una pérdida muy importante de experiencia y conocimiento. Para solventar el problema se sustituyó al empleado por un sistema experto que reproducía su experiencia y conocimiento.

Las diferencias básicas entre un sistema experto y un programa ordinario se pueden ver en la siguiente figura.

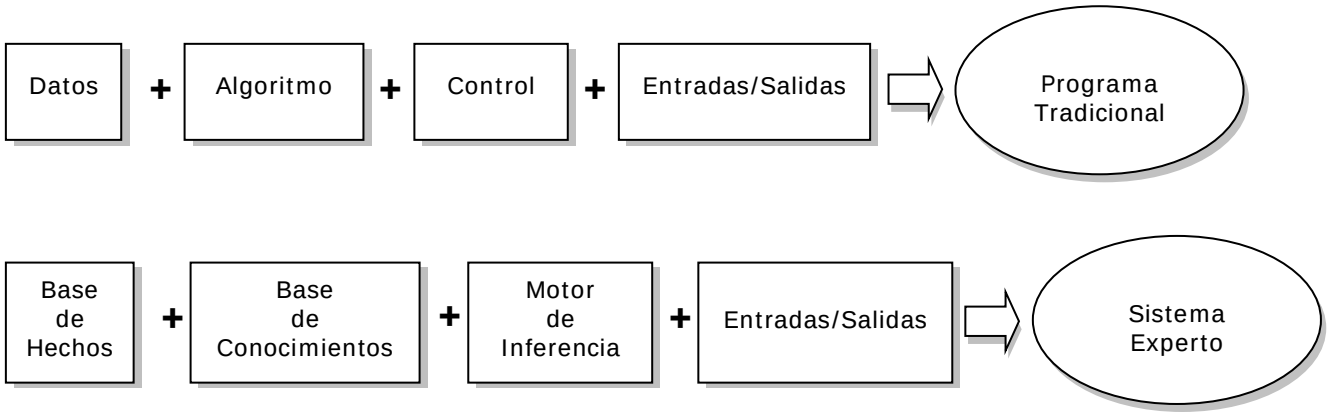


Figura 2.2: Flujos de información en los distintos sistemas

Existen varios tipos de sistemas expertos, pero los más destacables y extendidos son tres:

- Basados en reglas: trabajan mediante la aplicación de reglas, comparación de resultados y aplicación de las nuevas reglas basadas en situación modificada.

- Basados en casos: dan solución a determinadas situaciones nuevas basándose en las soluciones de problemas anteriores.

- Basados en redes bayesianas: modelo probabilístico multivariante que relaciona un conjunto de variables aleatorias mediante un grafo dirigido que indica explícitamente influencia causal. Su motor de actualización de probabilidades se basa en el Teorema de Bayes.

La forma más común en que se presenta un sistema experto es como un programa informático, con un conjunto de reglas (por lo general proporcionadas por el usuario del sistema), encargadas de analizar la información relativa a una clase específica de problemas y de proponer una o más acciones alternativas. El sistema experto podría también realizar un análisis matemático del problema. Un término relacionado con los sistemas expertos es “mago”, en inglés “wizard”. Un “mago” es un programa interactivo que ayuda a un usuario a resolver un problema. Originalmente este nombre se utilizó para programas que realizaban secuencias de búsqueda en bases de datos apoyándose para ello en las especificaciones del usuario. Sin embargo, algunos sistemas expertos basados en reglas se designan también con este término. Un sistema experto es muy eficaz cuando tiene que analizar una gran cantidad de información, interpretándola y proporcionando una recomendación a partir de la misma (SAMPER, 2009).

A continuación se va a explicar con más detenimiento cada una de las partes o capacidades que se mencionaron anteriormente y que constituyen un sistema experto.

Base del conocimiento

La base del conocimiento es una herramienta que pertenece tanto a la ciencia cognitiva como a la IA. En el campo de la ciencia cognitiva tiene que ver con cómo la gente almacena y procesa la información. En el de la IA, el principal objetivo es almacenar el conocimiento, de tal manera que los programas desarrollados computacionalmente puedan procesarlo y alcancen así la verosimilitud de la inteligencia humana. Las investigaciones hechas dentro del campo de la IA han tomado prestadas teorías de la ciencia cognitiva. Por tanto, hay técnicas de representación tales como marcos, reglas y redes semánticas que proceden de las teorías del procesamiento de la información por parte de los humanos. Puesto que el conocimiento es utilizado para alcanzar un comportamiento inteligente, el propósito fundamental de la representación del conocimiento es representarlo de tal manera que facilite la inferencia, es decir, obtener conclusiones a partir del conocimiento.

Algunas de las cuestiones más importantes que surgen en la representación del conocimiento dentro de la perspectiva de la IA son:

- ¿Cómo representan el conocimiento las personas?

- ¿Cuál es la naturaleza del conocimiento y cómo se representa?

- ¿El esquema de representación debería tratarse como un campo particular o como propósito general?

- ¿Cómo de expresivo es un esquema de representación?

- ¿Debería ser el esquema declarativo o procedimental?

Motor de inferencia

Está relacionado con la representación elegida para el conocimiento del experto y se utiliza para procesar ese conocimiento. Se puede utilizar la componente de adquisición de conocimiento del sistema experto para introducir como entrada las múltiples características conocidas para conseguir una buena técnica de inferencia.

Para obtener la mencionada técnica de inferencia adecuada, hay que tener en cuenta las siguientes ideas:

- Una buena técnica de inferencia es independiente del ámbito del problema.

- Para encontrar las ventajas de la explicación, transparencia del conocimiento y la reutilización de los programas en un nuevo ámbito, el motor de inferencia no debe contener información específica del problema.

- Las técnicas de inferencia pueden ser específicas de una tarea particular, tal como un diagnóstico de configuración de hardware. Otras técnicas pueden ser asignadas solamente a una técnica particular de procesamiento.

- Las técnicas de inferencia son siempre específicas para la estructura del conocimiento en cuestión.

Ventajas y limitaciones

Las principales ventajas de los sistemas expertos son:

- Permanencia: a diferencia de una persona experta un sistema experto no envejece, y por tanto no sufre pérdida de facultades con el paso del tiempo, si bien habría que actualizarlo con el paso del tiempo.

- Duplicación: una vez programado, el sistema experto lo podemos duplicar infinidad de veces.

- Rapidez: un sistema experto puede obtener información de una base de datos y realizar cálculos numéricos mucho más rápido que cualquier ser humano.

- Bajo coste: a pesar de que el coste inicial pueda ser elevado, gracias a la capacidad de duplicación éste se reduce.

- Entornos peligrosos: el sistema experto puede trabajar en entornos peligrosos o dañinos para el ser humano.

- Fiabilidad: Los sistemas expertos no se ven afectados por condiciones externas, un humano sí (cansancio, presión, etc.).

Las limitaciones de estos sistemas:

- Sentido común: para un sistema experto no hay nada obvio. Por ejemplo, un sistema experto sobre medicina podría admitir que un hombre lleva 40 meses en estado, a no ser que se especifique que esto no es posible.

- Lenguaje natural: con un sistema experto no se puede mantener una conversación para comunicarse.

- Capacidad de aprendizaje: cualquier persona aprende con relativa facilidad de sus errores y de errores ajenos, mientras que para un sistema experto esto es muy complicado.

- Perspectiva global: un experto humano es capaz de distinguir cuáles son las cuestiones relevantes de un problema y separarlas de cuestiones secundarias. Un sistema experto tiene más dificultades para realizar esta acción.

- Capacidad sensorial: un sistema experto carece de sentidos.

- Flexibilidad: un humano es sumamente flexible a la hora de aceptar datos para la resolución de un problema. El sistema experto los necesita de una determinada manera.

- Conocimiento no estructurado: un sistema experto no es capaz de manejar conocimiento poco estructurado, hoy por hoy.

Tareas de un sistema experto

- Monitorización

La monitorización es un caso particular de la interpretación, y consiste en la comparación continua de los valores de las señales o datos de entrada y unos valores que actúan como criterios de normalidad o estándares. En el campo del mantenimiento predictivo los sistemas expertos se utilizan fundamentalmente como herramientas de diagnóstico. Se trata de que el programa pueda determinar en cada momento el estado de funcionamiento de sistemas complejos, anticipándose a los posibles incidentes que pudieran acontecer. Así, usando un modelo computacional del razonamiento de un experto humano, proporciona los mismos resultados que alcanzaría dicho experto.

- Diseño

El diseño es el proceso de especificar una descripción de un artefacto que satisface varias características desde un número de fuentes de conocimiento.

El diseño se concibe de distintas formas:

- En ingeniería es el uso de principios científicos, información técnica e imaginación en la definición de una estructura mecánica, máquina o sistema que ejecute funciones específicas con la máxima eficiencia.

- El diseño industrial busca rectificar las omisiones de la ingeniería. Es un intento consciente de traer forma y orden visual a la ingeniería de hardware donde la tecnología no provee estas características.

Los sistemas expertos en diseño ven este proceso como un problema de búsqueda de una solución óptima o adecuada. Las soluciones alternativas pueden ser conocidas de antemano o se pueden generar automáticamente probándose distintos diseños para verificar cuáles de ellos cumplen los requerimientos solicitados por el usuario. Esta técnica es llamada “generación y prueba”, por lo tanto estos sistemas expertos son llamados de selección. En áreas de aplicación, la prueba se termina cuando se encuentra la primera solución; sin embargo, existen problemas más complejos en los que el objetivo es encontrar la solución más óptima.

- Control

Un sistema de control participa en la realización de las tareas de interpretación, diagnóstico y reparación de forma secuencial. Con ello se consigue conducir o guiar un proceso o sistema. Los sistemas de control son complejos debido al número de funciones que deben manejar y el gran número de factores que deben considerar; esta complejidad creciente es otra de las razones que apuntan al uso del conocimiento, y por tanto de los sistemas expertos.

Cabe aclarar que los sistemas de control pueden ser en lazo abierto, si en el mismo la realimentación o el paso de un proceso a otro lo realiza el operador, o en lazo cerrado si no tiene que intervenir el operador en ninguna parte del mismo.

- Reparación, corrección o terapia.

La reparación, corrección, terapia o tratamiento consiste en la proposición de las acciones correctoras necesarias para la resolución de un problema. Los sistemas expertos en reparación tienen que cumplir diversos objetivos, como la reparación lo más rápida y económicamente posible, orden de las reparaciones cuando hay que realizar varias y evitar los efectos secundarios de la reparación, es decir, la aparición de nuevas averías por la reparación.

- Simulación

La simulación es una técnica consistente en crear modelos basados en hechos, observaciones e interpretaciones en el ordenador, a fin de estudiar el comportamiento de los mismos mediante la observación de las salidas para un conjunto de entradas. Las técnicas tradicionales de simulación requieren modelos matemáticos y lógicos que describen el comportamiento del sistema bajo estudio. El empleo de los sistemas expertos para la simulación viene motivado por la principal característica de éstos, que es su capacidad para la simulación del comportamiento de un experto humano, lo cual es un proceso complejo.

En la aplicación de los sistemas expertos para simulación hay que diferenciar cinco configuraciones posibles:

- Un sistema experto puede disponer de un simulador con el fin de comprobar las soluciones y en su caso rectificar el proceso que sigue.

- Un sistema de simulación puede contener como parte del mismo a un sistema experto y por lo tanto éste no tiene que ser necesariamente de simulación.

- Un sistema experto puede controlar un proceso de simulación, es decir, que el modelo está en la base del conocimiento del sistema experto, y su evolución es función de la base de hechos, la base de conocimientos y el motor de inferencia, y no de un conjunto de ecuaciones aritmético-lógicas.

- Un sistema experto puede utilizarse como consejero del usuario y del sistema de simulación.

- Un sistema experto puede utilizarse como máscara o sistema frontal de un simulador con el fin de que el usuario reciba explicación y justificación de los procesos.

- Planificación

La planificación es la realización de planes o secuencias de acciones. Es un caso particular de la simulación. Está compuesto por un simulador y un sistema de control. El efecto final es la ordenación de un conjunto de acciones con el fin de conseguir un objetivo global. Los problemas que presenta la planificación mediante sistemas expertos son los siguientes:

- Existen consecuencias no previsibles, de forma que hay que explorar y explicar varios planes.

- Existen muchas consideraciones que deben ser valoradas o incluirles un factor de peso.

- Suelen existir interacciones entre planes de subobjetivos diversos, por lo que deben elegirse soluciones de compromiso.

- Trabajo frecuente con incertidumbre, pues la mayoría de los datos con los que se trabaja son más o menos probables pero no seguros.

- Es necesario hacer uso de fuentes diversas tales como bases de datos.

- Instrucción

Un sistema de instrucción realizará un seguimiento de un proceso de aprendizaje. El sistema detecta errores de una persona con conocimientos e identifica el remedio adecuado, es decir, desarrolla un plan de enseñanza que facilita el proceso de aprendizaje y la corrección de errores.

- Recuperación de información

Los sistemas expertos, con su capacidad para combinar información y reglas de actuación, han sido vistos como una de las posibles soluciones al tratamiento y recuperación de información.

Lo que diferencia a estos sistemas de un sistema tradicional de recuperación de información es que éstos últimos sólo son capaces de recuperar lo que existe explícitamente, mientras que un sistema experto debe ser capaz de generar información no explícita, razonando con los elementos que se le dan. Pero la capacidad de los sistemas expertos en el ámbito de la recuperación de la información no se limita a la simple recuperación. Pueden utilizarse para ayudar al usuario en selección de recursos de información, en filtrado de respuestas, etc. Un sistema experto puede actuar como un intermediario inteligente que guía y apoya el trabajo del usuario final.

Proyectos desarrollados

En este apartado se van a presentar algunos de los lenguajes de programación que sirven para desarrollar programas que sean sistemas expertos, y algunos trabajos llevados a cabo en este subcampo de la Inteligencia Artificial:

- MYCIN: sistema experto desarrollado en 1972 para el diagnóstico de enfermedades infecciosas (GAE, 2009).

- CADUCEUS: sistema experto finalizado a mediados de 1980. Su propósito era mejorar el MYCIN. Utilizaba un motor de inferencia para tratar la complejidad adicional de las enfermedades internas.

- Dendral: es un sistema experto desarrollado a mediados de los años 60. Es el primer sistema experto en ser utilizado para propósitos reales. Su objetivo era interpretar la estructura molecular (ALONSO, 2001).

- Dipmeter Advisor: sistema experto cuyo propósito era ayudar al análisis de los datos acumulados durante la exploración de aceite.

Programas

A continuación se van a presentar algunos de los lenguajes de programación que sirven para desarrollar sistemas expertos:

- ART: es un lenguaje de programación de propósito general utilizado en el desarrollo de sistemas expertos.

- CLIPS: es un software de dominio público para construir sistemas expertos. Esta herramienta está basada en el lenguaje de programación C.

- Drools: es un motor de reglas de encadenamiento basado en inferencia. Es un sistema de producción de reglas que utiliza una mejora del algoritmo Rete.

- JESS: es un lenguaje de programación desarrollado por Java y basado en CLIPS para la construcción de sistemas expertos.

- Prolog: lenguaje de programación lógico e interpretativo para desarrollar aplicaciones en Inteligencia Artificial.

2.6. Etapas en el desarrollo de un Sistema Experto

El desarrollo de un sistema experto con posibilidades de éxito debe programarse muy cuidadosamente.

El comienzo debe consistir en definir correctamente el problema a resolver. La experiencia demuestra que es una etapa muy importante en el diseño de un sistema experto y que se le dedica un tiempo muy por debajo del necesario.

La segunda etapa es la búsqueda de un experto humano o de los datos o experiencias.

Una vez que el problema está inicialmente definido, hay que buscar a un experto humano que esté en condiciones de resolverlo con posibilidades de éxito.

La tercera etapa es la de diseño del sistema experto en la que se incluyen la estructura para almacenamiento del conocimiento, la interfaz de usuario, el mecanismo de razonamiento, etc.

Con el resultado de la etapa anterior puede procederse ya al desarrollo de un prototipo y a la prueba del mismo, en un ciclo que se repite hasta que se consiguen los resultados buscados. A esta fase seguirá una de refinamiento y generalización en la que se van puliendo defectos e incluyendo nuevos casos no contemplados en el diseño inicial.

Como se puede observar, de las fases que componen el desarrollo de un sistema experto, las dos últimas son las que realmente se pueden observar cuando un usuario utiliza una herramienta de generación de sistemas expertos. Cabe por lo tanto decir que la conceptualización del conocimiento y su formalización son tanto o más críticas que las fases de implementación o puesta a punto, ya que de ellas (de la forma de tratar el conocimiento) depende el buen resultado del sistema experto.

2.7. Metodología para la construcción de Sistemas Expertos

A pesar de que no existe una metodología de concepción de sistemas expertos universalmente aceptada, los especialistas en la materia coinciden generalmente en admitir un esquema director en tres fases: en la primera, durante las discusiones con el experto humano, se intenta delimitar el problema y los modos de razonamiento puestos en práctica para su resolución. Luego, se ha de desglosar el formalismo de expresiones del conocimiento y el mecanismo de razonamiento apropiado, y finalmente, de acuerdo con los expertos, se describe la base de conocimientos, se ensaya y pone a punto con la ayuda de ejemplos.

Así pues, en primer lugar es necesario dialogar con un experto para tomar sus métodos y modos de actuación. Esta fase de análisis y estudio requiere la presencia física del experto humano y su participación activa. Cuando se habla de expertos, se designa una persona que conozca la materia por haberla practicado durante varios años. No es suficiente que tenga una idea teórica del problema que ha de resolver el sistema experto, también debe poseer una gran experiencia práctica.

Aunque la presencia de expertos, si bien es indispensable, no es una condición suficiente para la concepción de un sistema experto, es necesario, además, que el campo de conocimiento pueda ser fácilmente circunscrito y que el vocabulario mínimo que permita escribir todas las situaciones permisibles quede limitado a unos centenares de palabras.

Una vez reunidos expertos y técnicos en inteligencia artificial, se procede generalmente, a una primera entrevista, en el transcurso de la cual se circunscribe de manera precisa el tema. Se trata de plantear claramente el problema que tiene que resolver el sistema experto y localizar, en la actividad del experto, el lugar de la destreza.

Después de esta fase, es necesario escoger un experto que sea el experto maestro y al cual habrá de referirse durante toda la adquisición de la destreza. Se ha visto que durante la transferencia de la destreza hay que limitarse a un solo experto para evitar perderse en discusiones estériles.

Una vez hecha la elección del experto, hay que realizar la entrevista propiamente dicha. El primer objetivo es observar al experto a fin de detectar los nudos de razonamiento, identificar los objetos de trabajo y determinar el hilo o los hilos conductores. Para ello, es necesario hacer hablar al experto. Ha de poder expresarse lo más libremente posible y para

que así lo haga todos los medios son buenos: dibujos, gráficos, organigramas, magnetófono, etc.

Es importante observar sus tachaduras, sus retrocesos, sus tientos, sus dudas; en una palabra, todo aquello que, generalmente, no tendría lugar en una presentación académica, pero que podría ocultar una representación evolutiva del problema que hay que tomar en consideración.

Es también importante prestar una especial atención al lenguaje del experto. Para ello no es suficiente hacer un inventario de las palabras técnicas; es necesario, sobre todo, estudiar con precisión las palabras o las expresiones del lenguaje usual cuyo empleo es frecuente, para descubrir en ellas las indefiniciones y las ambigüedades. Estas son, a menudo, un indicio de conceptos operativos subyacentes.

Durante la entrevista con el experto se presenta otra dificultad importante que no deja de aparecer cada vez que se desea modelizar el razonamiento humano con ayuda de las técnicas de la inteligencia artificial. Obedece a que los hombres, para tratar problemas importantes, suelen proceder por etapas: en la primera, para reducir dificultades hacen un razonamiento general, frecuentemente inexacto. Luego, en vista de las contradicciones que aparecen, van afinando progresivamente su razonamiento hasta que ya no observan contradicción alguna.

Este tipo de razonamiento corresponde a lo que los especialistas denominan el razonamiento del sentido común. Para ponerlo en práctica hay que hacer intervenir simultáneamente unos razonamientos generales, a menudo falsos pero de alcance heurístico, unos conocimientos precisos cuya aplicación puede ser larga, y unas estrategias generales para llegar, con menos esfuerzo, a la solución del problema propuesto.

La modelización del sentido común es, por lo tanto, un punto crucial para la Inteligencia Artificial en el que, desafortunadamente, hasta ahora se ha fracasado en gran parte. Otro problema existente es la incertidumbre que existe tanto en el mundo real como en la heurística de los expertos. Se han utilizado unas técnicas que permiten afrontar estos problemas en unos casos determinados: los coeficientes de verosimilitud y la introducción de la nomonoticidad en los sistemas de producción.

Otra manera de abordar el razonamiento del sentido común es utilizar los sistemas no monotónicos. Se dice que un sistema es no monotónico cuando la retirada de un hecho, en una base de hechos coherente, puede llevar a una contradicción.

Pero, no obstante, la gestión de la no monotonicidad es extraordinariamente laboriosa y si no se tiene el cuidado suficiente conduce a una explosión combinatoria.

Teniendo en cuenta el estado de la técnica, no puede pretenderse resolver el problema total del sentido común. Por tanto, es necesario tratarlo caso por caso identificando los tipos de razonamientos aplicados por el experto, es decir, utilizando lo que se ha llamado el nivel estructurante mediante el despliegue y la observación de la totalidad del razonamiento del

experto, especialmente las distintas interacciones a las que es conducido y las justificaciones de estas interacciones.

Por lo demás, la definición de los modos de razonamiento no basta para esclarecer totalmente el nivel estructurante. Por otra parte, es necesario determinar la complejidad del problema a resolver. Gracias a ella, será posible definir la representación de los conocimientos necesarios para la situación determinada, es decir, es preciso estructurar la base de hechos.

A pesar de que no existe ninguna tipología sistemática que permita clasificar los problemas en función de su complejidad, se ha conseguido desglosar unas clases de complejidad, entre las cuales se distinguen habitualmente:

- El diagnóstico directo.
- La resolución de problemas.
- La planificación.

El diagnóstico directo es la más sencilla de estas clases de complejidad. Aquí se trata solamente de caracterizar una situación con ayuda de un descriptor. Por ejemplo:

enfermedad = angina

es un diagnóstico. Para representar el conocimiento, la mayoría de las veces puede bastar un formalismo equivalente a la lógica de las proposiciones: es el formalismo más elemental que los especialistas denominan como lógica de orden cero.

En el caso de la resolución de problemas, el enunciado ha de ser especificado por un conjunto de ecuaciones, algunas de las cuales han de ser parametrizadas con variables. Hallar una solución a un problema consiste en encontrar una serie de transformaciones válidas con ayuda de las cuales es posible aislar las incógnitas y hallar el o los valores que es necesario asignarles para satisfacer todas las condiciones del enunciado. Para ello hay que hacer intervenir una representación de los conocimientos equivalentes a la lógica de los predicados de primer orden.

Por último, cuando hay que tratar problemas del tipo planificación de tareas, es necesario enfrentarse a una gran complejidad, que tiene un doble origen: por una parte, el problema sólo es especificado por un conjunto de restricciones que hay que optimizar; por otra parte, el universo evoluciona en el transcurso de la ejecución de las distintas tareas. En la medida en que el universo evoluciona después de cada tarea, hay que memorizar las acciones ya ejecutadas para evitar secuencias infinitas de acciones. Para superar esta dificultad, es necesario introducir unas reglas generales referentes a las acciones ya realizadas.

Después de la fase de esclarecimiento del nivel estructurante, mediante el cual se ha desglosado ya un formalismo de la representación del conocimiento, a continuación, se debe utilizar este formalismo para escribir el contenido de la base de conocimientos.

Esta última etapa puede, a su vez, dividirse en dos fases:

• En un primer momento, se ha de definir un lenguaje de expresión. Es la dilucidación del nivel conceptual.

• Posteriormente, este lenguaje se emplea para transcribir el conocimiento del experto. Es la dilucidación del nivel cognoscitivo.

2.8. Formas de representar el conocimiento

La representación del conocimiento puede hacerse de muy diversas maneras. La elección del sistema de representación depende del tipo de problema que se pretenda resolver con el sistema experto, por lo que no puede darse un sistema que sea óptimo en todos los casos.

2.8.1. Redes Semánticas

Una red semántica es una colección de objetos, también llamados nodos, unidos mediante arcos o enlaces formando una red.

Los nodos representan objetos y los enlaces, relaciones entre los objetos.

Partiendo de una red semántica resulta fácil responder a multitud de preguntas.

De las redes semánticas cabe destacar la cantidad de información que es posible almacenar en una red de este tipo.

En la figura 2.3 se muestra un ejemplo de red semántica:

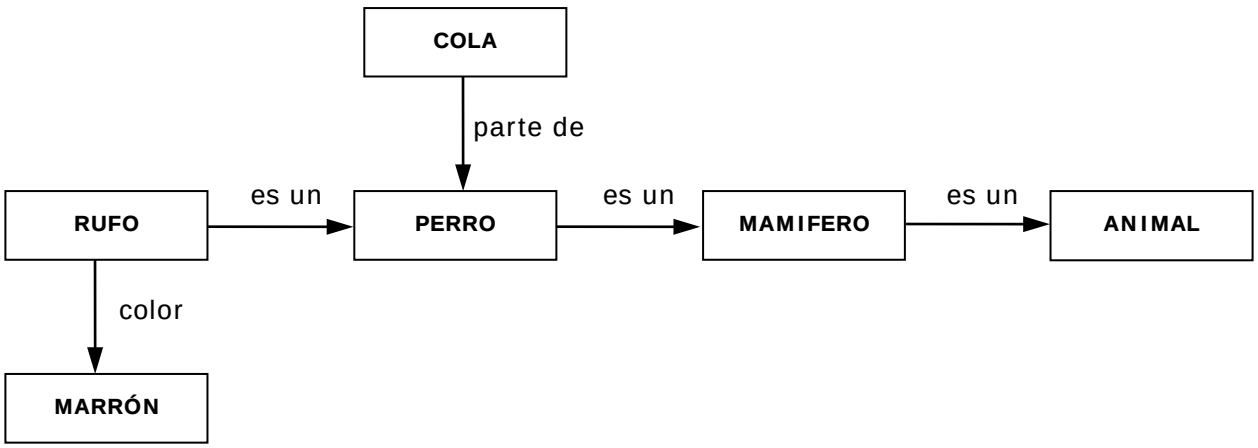


Figura 2.3. Ejemplo de red semántica.

2.8.2. Ternas

Las ternas son un caso especial de redes semánticas.

Existen tres nodos: objetos, atributos y valores y se suprimen los enlaces, que resultan triviales.

A veces a las ternas se les añade un factor de certeza que sirve para medir la confianza que se tiene de que la terna sea correcta.

En algunos casos se omite el atributo y se trabaja sólo con parejas objeto-valor.

La figura 2.4 muestra un ejemplo de terna y la figura 2.5 muestra esa misma terna en forma de red semántica:

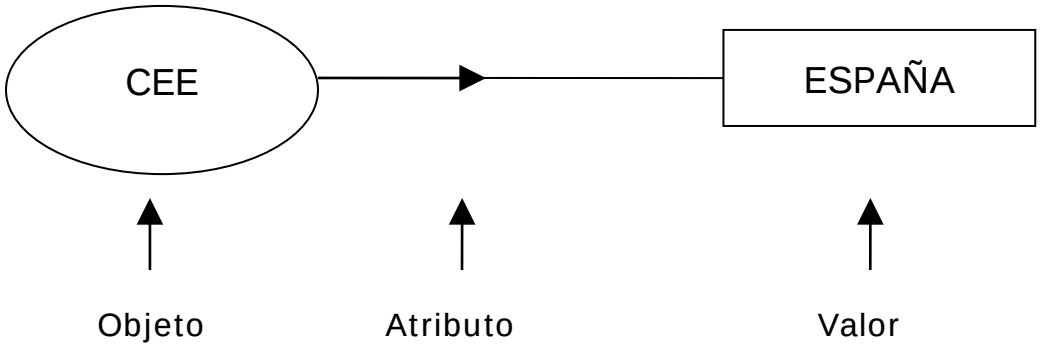


Figura 2.4. Ejemplo de terna.

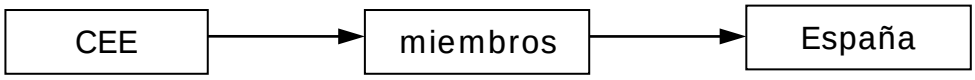


Figura 2.5. Terna anterior expresada en forma de red semántica.

2.8.3. Reglas

Otra forma de almacenar el conocimiento es mediante reglas. Las reglas se utilizan para representar relaciones y constan de dos partes: la premisa y la conclusión.

La premisa consta del condicional "SI" y de una expresión lógica.

La conclusión consta del adverbio "ENTONCES" y de una acción.

Si tras evaluar la expresión lógica de la premisa de una regla ésta resulta cierta, entonces se hace que la expresión lógica de la conclusión sea cierta.

Un ejemplo podría ser:

SI fecha_cheque = correcta y

firma_cheque = existente y

cantidades = concuerdan

ENTONCES

cheque = completo

A veces la reglas no son siempre ciertas o válidas y se les asocia al igual que a las ternas, un factor de certeza.

2.8.4. Marcos

Los marcos pueden ser considerados también como casos particulares de redes semánticas.

Un marco incluye toda la información existente sobre un objeto.

Esta información puede ser de tipo declarativa o descriptiva o del tipo indirecta, es decir, puede dar directamente las características del objeto o dar procedimientos o reglas para determinarlas.

Es la forma de almacenamiento más utilizada.

Un ejemplo podría ser:

Marco: Automóvil

Especialización de: Vehículo campero

Carrocería: Acero

Ventanas: Vidrio

Combustible sobrante:

Rango: (vacío, 1/4 tanque, 1/2 tanque, lleno)

Caso contrario: ninguno

SI_NECESARIO: verifique el indicador de combustible

Tipo de accidente:

Rango: (golpe-pequeño, severo, total)

Caso contrario: Ninguno

SI_NECESARIO: llamar agente de seguros

2.8.5. Expresiones lógicas

La lógica es una disciplina que está relacionada con la validez de los argumentos por dar métodos para determinar si unas conclusiones pueden deducirse correctamente a partir de los hechos.

Dentro de la lógica, la noción de argumento verdadero juega un papel importante. Un argumento es verdadero si y sólo si al ser todos sus supuestos verdaderos entonces sus conclusiones también lo son.

Hay varias notaciones o sistemas de lógicas, pero las más comunes son: la lógica proposicional y la lógica de predicados.

La lógica proposicional es un sistema de lógica común en el que las proposiciones son expresiones que pueden ser verdaderas o falsas y se ocupa de las expresiones compuestas.

Existen diferentes reglas para propagar la veracidad de las expresiones dependiendo de los conectivos. Si una proposición X es verdadera y otra Y es falsa, la expresión compuesta X e Y es falsa, mientras que la expresión X o Y es verdadera. Otras reglas permiten las inferencias. Si es sabido que X es verdadero y X implica Y, se puede decir que Y es verdadero.

El cálculo de predicados es una extensión de la anterior. La unidad fundamental en la lógica de predicados es un objeto. Las expresiones acerca del objeto se llaman predicados.

2.9. Incertidumbre

Una de las dificultades más importantes en un sistema experto es la capacidad para abordar efectivamente la información imprecisa, incompleta y algunas veces con incertidumbre.

Existen muchas clases diferentes de incertidumbre que son comunes en dominios de expertos como son:

- Conocimiento incierto. Por ejemplo, el experto podría saber solamente que cierto conjunto de evidencias implicaría probablemente una determinada conclusión.

- Datos inciertos. Por ejemplo, cuando intentamos deducir una causa específica a partir de un efecto observado, pudiéramos haber confiado en los cuestionables resultados de la prueba.

- Información incompleta. Por ejemplo, podemos tomar tales decisiones en el curso del procesamiento de la información adquirida en forma incremental.

- Azar, ya que algunos dominios son esencialmente azarosos.

Los siguientes apartados describen, un poco por encima, métodos para manejar la incertidumbre:

2.9.1. Razonamiento Basado en Información Parcial

Los razonamientos basados en lógica de predicados son conceptualmente elegantes e intelectualmente atractivos en razón de su rigurosidad y precisión.

Una vez establecida la verdad, siempre es verdadera.

La lógica de predicados es un sistema de razonamiento monotónico, que significa "moverse solamente en una dirección". Da la idea de un proceso de razonamiento que se mueve en una sola dirección, aquella a la que continuamente se agrega la verdad.

Los procesos de razonamiento que se han de aplicar a problemas prácticos no estructurados deben tener en cuenta por lo menos lo siguiente:

• La información disponible con frecuencia está incompleta.

• Las condiciones cambian en el tiempo.

• Existe la necesidad de lograr una adivinación eficiente, pero posiblemente incorrecta, cuando el razonamiento llega a un callejón sin salida.

2.9.2. Razonamiento no monotónico

Un sistema de razonamiento no monotónico incluye, por lo general, un conjunto de premisas que se mantienen como verdades inmutables.

Adicionalmente a las premisas, el sistema conserva una colección de creencias tentativas: porciones de conocimientos que son explícitamente reconocidos como posibles incorrecciones porque son supuestos o creencias deducidas de suposiciones.

Estos sistemas revisan las creencias cuando se observa o se deduce nuevo conocimiento.

Un sistema de razonamiento no monotónico es un sistema especialmente valioso para los dominios de solución de problemas, como

planificación y diseño, que requieren un número grande de supuestos tentativos basados en información parcial.

Por ejemplo cuando tomamos la decisión de tomar un avión, probablemente asumimos que el piloto es competente y que el aparato está en perfectas condiciones para volar.

2.9.3. Razonamiento con bases en probabilidades

Las técnicas de probabilidades se han empleado ampliamente en un intento de cuantificar la incertidumbre. El atractivo de las probabilidades se basa parcialmente en el hecho de que se ha establecido sobre una base matemática sólida.

P(E) es la probabilidad de que ocurra el evento E. En la mayoría de los casos el valor de P(E) se determina por medio de análisis estadístico.

Las probabilidades tienen un valor de 0 a 1, donde 1 representa el conocimiento absoluto de que E ocurre y 0 el conocimiento absoluto de que E no ocurre.

La suma de las probabilidades de todos los posibles resultados para un evento dado debe ser igual a 1.

Existe un serio problema en la utilización de probabilidades para tratar la incertidumbre en un sistema experto y es que, aunque una persona sea experta en un dominio, puede serle muy difícil estimar exactamente las probabilidades de un evento.

Dentro de este tipo de tratamiento de la incertidumbre el sistema de Bayes es ampliamente utilizado, sobre todo en situaciones complicadas y en dominios de diagnósticos.

La teoría de la probabilidad se ha aplicado exitosamente en varios sistemas expertos por ejemplo PAGE-1 y principalmente en el sistema PROSPECTOR.

2.9.4. Factores de certidumbre

Un factor de certidumbre es una cuantificación subjetiva del juicio y de la intuición de un experto. Es un valor numérico que expresa el punto al que, basándonos en un conjunto de evidencias, debemos aceptar una conclusión determinada.

Un factor de certidumbre con un valor 1, indica la creencia total, mientras que un valor -1 indica lo contrario.

Es un mecanismo informal para medir el grado al cual, basándonos en la presencia de un conjunto dado de evidencias, creemos o no en la conclusión dada.

El concepto de factor de certidumbre se desarrolló para MYCIN, debido a la dificultad de estimar en forma precisa las probabilidades a priori y condiciones que se requieren para aplicar el teorema de Bayes.

2.9.5. Razonamiento difuso

El razonamiento difuso se diseñó específicamente para tratar la inexactitud que está presente en el conocimiento empleado por los expertos.

Dada una comprensión correcta de la definición de conjunto, es posible determinar si un candidato es o no miembro del conjunto, pero cabe destacar el problema de la dificultad de desarrollar exactamente definiciones de conjunto para muchos conceptos y mecanismos de clasificación que el hombre emplea.

Dentro del razonamiento difuso debemos hablar del concepto de conjunto difuso, que es una clase de elementos con límites débilmente definidos.

Para identificar un miembro de un conjunto difuso, asociamos un grado de membresía a cada uno de los elementos, entre 0 y 1 que indica el punto hasta el cual un elemento es miembro del conjunto. Un grado 1 indica que el elemento es miembro mientras que un grado 0 indica que no lo es.

Los grados de membresía se asignan de forma subjetiva de acuerdo con el contexto.

2.9.6. Distintos tipos de Sistemas Expertos según sus aplicaciones

Se pueden distinguir varios tipos generales de sistemas expertos según sea la misión que han de realizar éstos. Así, por ejemplo, se pueden encontrar:

• Sistemas de interpretación. A partir de unos datos de entrada realizan la interpretación de éstos. Entre estos sistemas podemos nombrar: análisis de imagen, comprensión (traducción) de idiomas, aclaración de estructuras químicas, etc.

• Sistemas de predicción. Su misión es la de buscar unas consecuencias verosímiles a partir de una situación dada (datos de entrada).

• Sistemas de diagnosis. A partir de una serie de fallos observables (mal funcionamiento) indican a qué puede ser debido ese fallo. Dentro de este tipo podemos nombrar los sistemas expertos de medicina, electrónicos, mecánicos, etc.

Estos métodos pueden generalmente usar dos caminos distintos:

1. Usar una tabla de asociaciones entre conductas y diagnósticos.

2. Combinar el conocimiento del sistema diseñado con el conocimiento de sus posibles imperfecciones.

• Sistemas de diseño. Su labor es la de proporcionar unas determinadas configuraciones de objetos que satisfacen una determinada situación. Entre estos podemos destacar el diseño de edificios, diseño de presupuestos, diseño de circuitos, etc.

• Sistemas de planificación. Están especializados en proyectar distintos planes de actuación. Se pueden mencionar los sistemas dedicados a movimientos de robots, comunicaciones, planificación de experimentos, planificación militar, etc.

• Sistemas instructores. Son los que comparan las observaciones de la conducta de un sistema con el carácter más notable o vulnerable. Un ejemplo de estos sistemas son los de centrales nucleares o trafico aéreo.

• Sistemas descubridores de errores. Prescriben remedios para un mal funcionamiento.

Estos sistemas se componen a su vez de subsistemas de planificación, diseño y predicción. Y han de ser capaces de crear especificaciones o recomendaciones para rectificar el diagnóstico de un problema.

• Sistemas de reparación. Han de ser capaces de suministrar un remedio para un determinado problema. Estos sistemas incorporan subsistemas de descubrimiento de errores, planificación de ejecución de capacidades. Un ejemplo de ellos son los dedicados a la reparación de automóviles, de aviación y de ordenadores.

• Sistemas de control. Están formados por subsistemas de interpretación, predicción, reparación e instrucción de la conducta del sistema. Se puede decir que estos sistemas interpretan la situación, predicen el futuro, diagnostican las posibles causas de problemas con anticipación y formulan planes para remediarlo.

Es evidente que un sistema experto, como ya se ha visto, puede contar con varios tipos de subsistemas. Si bien, a mayor número de subsistemas el sistema es más complejo, tanto de llevar a cabo, como de utilizar una vez hecho.

3. DESARROLLO DE UN SISTEMA EXPERTO

3.1. ¿Cómo construir un Sistema Experto?

Puesto que un sistema experto es un programa de ordenador, para realizarlo es necesario utilizar un lenguaje de programación.

Habrá unos lenguajes mejores que otros o incluso se pueden utilizar herramientas más evolucionadas que los lenguajes básicos, aunque estas, estén basados en ellos.

Una de las principales razones para el rápido crecimiento de los sistemas expertos, es el conjunto enriquecido de herramientas poderosas de desarrollo.

La selección de una herramienta correcta o de un conjunto de herramientas, es una decisión clave en el desarrollo de un sistema experto.

Las herramientas para sistemas expertos son valiosas por las siguientes razones:

1. Proporcionan ambientes enriquecidos para el desarrollo de cualquier software. Los componentes de este ambiente incluyen cosas como editores de estructuras, paquetes potentes para depuración y rastreo, multiventanería, gráficas y facilidades de señalamiento.

2. Incluyen ayudas específicas para prototipos rápidos. Estas facilidades incluyen cosas como compiladores incrementales, registros históricos de cambio detallados y control automático de versiones.

3. Se puede evitar gran parte de la definición de modelos de consulta, representación del conocimiento y paradigma de inferencia porque estas facilidades están incorporadas en las herramientas. Esto tiene grandes ventajas. Permite a alguien que esté menos familiarizado con conceptos de inteligencia artificial (como el experto en algunos casos), desarrollar un sistema. Provee una base sólida para una rápida captación del conocimiento y un rápido desarrollo del sistema y elimina el trabajo requerido para construir el software básico.

4. En muchos casos, una herramienta para sistemas expertos, puede proporcionar una amplia ayuda en algunas áreas específicas del desarrollo del sistema, incluyendo parte del dominio del conocimiento, en la adquisición del conocimiento, o en áreas tales como verificación del sistema.

Existen diferentes tipos de sistemas expertos y estas diferencias tienen su origen en los distintos problemas que pueden abordar, como ya hemos visto en el capítulo anterior.

Cada una de estas familias de problemas puede acomodarse a métodos de control e inferencia parecidos.

En función de esta posibilidad se diferencian los distintos tipos de herramientas básicas en la creación de sistemas expertos.

3.2. Lenguajes de propósito general

La herramienta más elemental para el desarrollo de sistemas expertos es un lenguaje de propósito general. Lisp ha sido, y continua siendo, el lenguaje más ampliamente empleado para el desarrollo de sistemas expertos. Proporciona muchas características que facilitan las tareas para la construcción de cualquier sistema de procesamiento simbólico. Más aún, Lisp se está volviendo más popular para la programación convencional con el surgimiento del COMMON LISP.

El lenguaje de programación Prolog está ganando popularidad, aunque su empleo como lenguaje para el desarrollo de sistemas expertos ha sido un poco menor que Lisp. Prolog también es un lenguaje simbólico de propósito general, y por tener un método incorporado de búsqueda es un poco más específico que Lisp.

3.2.1. LISP

Lisp es abreviatura de "list programming, list procesing lenguaje" se ha hecho popular entre los especialistas en inteligencia artificial porque fue diseñado para procesar (y formar asociaciones entre) símbolos tales como palabras.

Se diferencia en esto de la mayoría de los lenguajes de programación, concebidos para realizar principalmente cálculos numéricos. En realidad, otros lenguajes podrían usarse para cierta clase de procesamiento simbólico, pero Lisp y Prolog son enormemente más eficientes en esta tarea que los orientados a cálculo numérico, como por ejemplo Fortran.

Los programas de inteligencia artificial destacan por sus enormes requerimientos de memoria y ciclos de máquina, especialmente durante el desarrollo. Para acelerar el proceso de desarrollo, los investigadores, gradualmente, construyen un entorno de herramientas de programación, tales como programas depurados y sofisticadas interfaces gráficas.

El entorno resultante de Lisp sobrepasa en mucho a aquellos lenguajes de programación de tipo numérico.

Más aún, las herramientas de programación por sí mismas no eran una solución suficiente para el desarrollo de la inteligencia artificial, y a mediados de los años 70 se pusieron en marcha proyectos independientes en el Massachusetts Institute of Technology (MIT) y en Palo Alto Research Center, de la empresa Xerox, con el objetivo de construir hardware específicamente adecuado a Lisp y a su entorno de desarrollo. La disponibilidad de ordenadores comerciales diseñados específicamente para explotar el entorno de desarrollo de Lisp, llevó la inteligencia artificial a un escalón más cerca del mercado consumidor, pero las primeras máquinas eran caras y dedicadas solamente a investigadores de laboratorio.

Desde entonces, se presentaron modelos nuevos y más económicos, pero la conquista del mercado de los usuarios finales no se consiguió debido principalmente a la revolución de la microinformática, que ha producido la desaparición de estos ordenadores por su alto precio.

El lenguaje Lisp lo creó John McCarthy hacia 1960 para conseguir una programación no-numérica.

Su nombre indica que la estructura fundamental de Lisp son listas. Estas listas están formadas por una combinación de símbolos abstractos llamados átomos. El átomo es el elemento más pequeño de Lisp. No se divide en partes. El átomo tiene un nombre, por ejemplo "casa" y puede tener propiedades o atributos relacionados con él, el más importante de todos es el llamado valor.

Una lista en Lisp está formada por un conjunto de átomos entre paréntesis. Aunque las listas también pueden estar formadas por otras listas.

Las funciones básicas de Lisp, a partir de las cuales se desarrollan todas las demás son:

CAR. Proporciona el valor del primer elemento.

CDR. Da el valor de la lista sin el primer elemento.

CONS. Junta dos expresiones para crear una lista.

ATOM. Analiza si una estructura es lista o átomo.

EQUAL. Proporciona el valor verdadero si son iguales dos expresiones.

3.2.2. PROLOG

Prolog es la abreviatura de "programing in logic".

Los inconvenientes de la programación imperativa, con la que han funcionado las cosas durante cuarenta años, son numerosos. Entre otros, el más evidente es que sólo puede hacerse un programa, según este principio, si se ha puesto a punto claramente un método de resolución, un algoritmo, adaptado al problema abordado o a un problema más general. Así, no se podrá realizar un programa para la resolución de las ecuaciones de segundo grado de una incógnita, si no se traduce en un lenguaje de programación imperativo el célebre algoritmo aprendido en la escuela, o utilizando un programa más general de resolución de ecuaciones polinómicas que planteará el mismo problema pero a otra escala. Cabría esperar pues, que una máquina calificada por algunos de inteligente pueda al menos ayudar al descubrimiento del propio método. En general, esto no es así.

El segundo inconveniente también de la programación imperativa, es que obliga a realizar un programa o una parte del programa diferente para cada tipo de preguntas susceptibles de ser planteadas por el futuro usuario,

o bien, como ya se ha visto, resolver el problema general correspondiente al cálculo de la respuesta a todas estas preguntas. Un programa que gestiona un fichero de personas escrito para responder a las preguntas de la forma: ¿Qué edad tiene X? o ¿Cuáles son las personas que tiene edad Y?, es tratado de distinta forma en un lenguaje de propósito general que por un lenguaje convencional. Sin embargo, todo el mundo cree que estos dos problemas están estrechamente ligados, y que definiendo el primero, se tiene implícitamente definido el segundo.

Las limitaciones de la programación imperativa se hacen sentir muy fuertemente en la actualidad, puesto que la realización de programas se ha convertido en la actividad más importante de la informática.

El coste de los equipos no cesa de disminuir, pero la escritura de los programas continúa siendo muy costosa, especialmente porque se exige una habilidad cada vez mayor y los programas son cada vez más grandes. Efectivamente, en este campo, la dificultad aumenta en función del tamaño de una forma que se aproxima más a la exponencial que a la recta.

Estas dificultades son particularmente sentidas por los investigadores en inteligencia artificial, que por definición deben realizar sistemas capaces de responder a una gran variedad de preguntas. Efectivamente, es muy pesado escribir, en cada caso, el algoritmo que utiliza para responder a una pregunta, incluso cuando los principios generales que deben emplearse están claramente definidos.

Una solución consiste en realizar programas muy generales y, sobre todo, muy "reutilizables" de una máquina a otra, naturalmente, pero también de una aplicación a otra.

Esta idea se aplica, por lo menos en teoría, en los sistemas expertos, puesto que un mismo "sistema experto esencia" está capacitado para aprender tan bien el arte de diagnosticar una ictericia, como las averías que se producen en un motor diesel.

En cierta manera, es sobre este principio sobre el que se basan los lenguajes de programación declarativos. La máquina que ejecuta los programas escritos en Prolog (una máquina virtual simulada, en ausencia de una verdadera máquina Prolog) con un ordenador clásico, incorpora un formalismo de representación de los conocimientos reforzado con un mecanismo general de resolución de problemas.

En Prolog, la información declara un cierto número de hechos que corresponden a una problemática más que a un problema, y la máquina Prolog pone en marcha un mecanismo de inferencia lógica para buscar la respuesta a cualquier pregunta planteada. Si el programa contiene suficientes conocimientos para que puedan deducirse soluciones, se dan al usuario.

En un lenguaje clásico, el programador indicaría, por ejemplo, el siguiente algoritmo:

"Restar la fecha de nacimiento, de la fecha de hoy, para obtener la edad y después, si esta es inferior a 18, deducir el 30% de la tarifa para obtener el precio, si no...".

En Prolog, este mismo programador declararía: "El precio para los menores es igual a la tarifa disminuida en un 30%; los menores son las personas cuya edad es inferior a 18; la edad de un persona es la diferencia entre al fecha del día y su fecha de nacimiento; en este orden o en cualquier otro, incluso en una forma algo más alejada del lenguaje corriente.

De hecho, desde hace mucho tiempo se está intentando introducir esta nueva forma de programar, y es necesario retroceder veinte años para asistir al nacimiento del Prolog en la facultad de ciencias de Luminy en Marsella.

El objetivo era entonces integrar en un lenguaje de programación un reciente desarrollo de la lógica matemática: el principio de resolución, propuesto por Alan Robison, que es un método de demostración automática de teoremas.

Este principio permite construir de manera sistemática una demostración con la ayuda de deducciones elementales que se basan en la aplicación de una sola y única regla de deducción, mientras que los métodos clásicos utilizan varias, entre las que hay que escoger según las circunstancias. Gracias a este principio, es posible crear un lenguaje de programación capaz de "razonar" según la lógica de primer orden, la que se emplea cuando se deduce que "Sócrates es mortal", a partir de la premisa, "todo hombre es mortal" y "Sócrates es un hombre".

En Prolog, el problema de Sócrates se describe en dos reglas:

mortal(X) > hombre(X)

y

hombre(Sócrates)

La flecha debe interpretarse como la expresión "si" y no como la expresión "implica", la ausencia de una expresión a la derecha de la flecha en la segunda regla significa que lo que se ha expresado a la izquierda es un hecho comprobado en todas las circunstancias. A la pregunta "¿mortal (X)?" Prolog responde inmediatamente: "(X=Sócrates)".

El origen de Prolog se encuentra al final de una larga tradición lógica. Sin embargo, la pura y simple transposición de los principios de partida, a veces se reveló contradictoria con los imperativos de eficacia de la informática.

En el momento de realizar un programa capaz de comprender el nuevo lenguaje, ha sido necesario repercutir estas contradicciones sobre la propia teoría de Prolog, lo que ha conducido a la elaboración de un nuevo modelo teórico autónomo, y que se beneficia de diez años de experiencia y simplificación.

La mejor manera de detallar el desarrollo de un programa Prolog en una máquina, es idealizar esta en una máquina abstracta reducida a los únicos mecanismos pertinentes.

El objetivo de la ejecución de un programa Prolog, es contestar a una pregunta representada por un término. Las respuestas que se calculan son los conjuntos de limitaciones elementales que conducen a las variables del término, con las que este representa hechos especificados por las reglas del programa. El objetivo inicial (el término que representa la pregunta) se reemplaza progresivamente por nuevos conjuntos (los términos que proceden de las partes derechas de las reglas aplicadas) al mismo tiempo que se acumulan las condiciones impuestas por la aplicación de las reglas.

Al principio solo hay un objetivo, la pregunta y ninguna condición. Mientras queden objetivos que alcanzar se intenta aplicar cada una de las reglas del programa. Cuando una de ellas es aplicable, es decir, que se reemplaza el primero de los objetivos por alcanzar por el miembro de la derecha de la regla, se añaden las condiciones correspondientes al conjunto de las mismas. Entonces se pasa a la etapa siguiente, puesto que ahora debe hacerse lo que acaba de realizarse para todos los objetos que quedan (los que se acaban de sustituirse anteriormente).

Cuando ya no quedan más objetivos que alcanzar, en general el trabajo no ha terminado. Todavía hay que volver atrás, hacia las etapas anteriores, para reemprender la prueba sistemática de las reglas que todavía no se han intentado aplicar. Finalmente, cuando se ha vuelto a la etapa inicial y ya no quedan más reglas que probar, el trabajo está terminado.

3.2.3. Otros

Hay también una tendencia creciente hacia el empleo de lenguajes más convencionales, para el desarrollo de sistemas expertos. Varios sistemas expertos importantes se han desarrollado en Fortran y se han construido un gran número de ellos en lenguaje C.

Existen varias desventajas para utilizar tales lenguajes. La más notoria, es su carencia de soporte para el procesamiento simbólico y para la administración de memoria automática.

Típicamente, se requiere más tiempo para construir un sistema experto, empleando un lenguaje convencional, que usando un lenguaje simbólico. Otra desventaja es el tipo de desarrollo y el tiempo de mantenimiento.

Sin embargo, existen grandes ventajas:

• C por lo general, correrá más rápido que Lisp en un equipo de propósito general, en algunos casos, mucho más rápido.

• C también está más extensamente disponible y tiene un soporte general amplio.

• Existe mayor disponibilidad de programadores especializados en C que en Lisp .

• El uso de C, posibilita al sistema final para integrarse de forma fácil, con el software externo existente.

Para alcanzar lo mejor de ambos mundos, muchos desarrolladores implementan el sistema considerando el empleo de un lenguaje simbólico y convierten la versión final, a un lenguaje convencional para su entrega.

Esto, en principio, es muy aplicable cuando se anticipa que el sistema requerirá de cambios relativamente pequeños después de la entrega, ya que anteriormente hemos comentado la gran desventaja que presentan los lenguajes simbólicos, a la hora de realizar el mantenimiento.

3.3. Estructuras generalizadas para Sistemas Expertos

El principal desarrollo en herramientas específicas de sistemas expertos, fue la introducción de las estructuras generalizadas (Shell) para sistemas expertos que representan el esqueleto del antiguo sistema experto.

Sobre un sistema experto ya construido, se recoge el mayor número de elementos compatibles con otro. Por ejemplo, la forma de representar el conocimiento, el control, la inferencia.

En los sistemas shell, la forma óptima de representar el conocimiento es mediante reglas, ya que permite cambiar el significado de las mismas sin cambiar la estructura.

El hecho de utilizar un esqueleto de un sistema experto sobre otro puede representar muchos problemas, sobre todo si se requieren tareas distintas a las concebidas para el sistema original.

Los principales problemas pueden ser:

• El armazón del antiguo sistema puede resultar inapropiado para la nueva tarea a realizar.

• La estructura de control puede no encajar perfectamente con la forma de realizar las inferencias que se desee.

• El lenguaje antiguo puede resultar inapropiado.

• Puede existir en el sistema un conocimiento previo sobre el control que no se necesite.

El valor de un shell para un sistema experto está directamente relacionado con el grado en que las características del dominio se parecen a las características esperadas por el modelo interno del shell.

La mayoría de las herramientas expuestas en este apartado son capaces de razonar únicamente con reglas de producción, si bien algunas de

ellas no se encuentran en el mercado, otras que están ahora en el mercado se basan fundamentalmente en esta modelización del conocimiento.

3.3.1. EMYCIN

El primer ejemplo de tales herramientas es Emycin que salió del proyecto Mycin. Mycin fue desarrollado usando Lisp en vez de una herramienta específica para sistemas expertos. Después de que se terminó el desarrollo, se reconoció que el sistema podría verse como dos componentes separados: el sistema básico (representación del conocimiento, motor de inferencia) y el conocimiento específico del dominio, es este caso diagnóstico médico.

A partir de esta observación, nació Emycin. Emycin es una estructura generalizada (Shell) en el sentido que es un sistema experto, al cual se le quitó el conocimiento del dominio específico. Este shell luego se puede aplicar para la construcción de sistemas expertos diferentes, para dominios similares.

Emycin se basa en reglas con una estrategia de inferencia con encadenamientos hacia atrás y se diseñó para tareas de consulta de diagnóstico. Se ha probado que es muy útil para problemas que se parecen a éste modelo básico, pero no ha sido muy útil para los dominios que no se parecen (tareas de diseño que requieren forma de planeamiento).

3.3.2. EXPERT

Expert es un sistema de programación desarrollado para la construcción de modelos de consulta basados en problemas de clasificación. Ha sido usado primordialmente para desarrollar modelos de diagnósticos en medicina, especialmente en oftalmología, endocrinología y reumatología.

Un modelo Expert, selecciona las hipótesis apropiadas por interpretación de las reglas y observaciones.

Cada modelo consta de hipótesis y reglas de decisión. Las hipótesis son las conclusiones que el modelo puede deducir.

Expert usa tres tipos de reglas: FF, FH, HH

Una regla de tipo FF es:

si HECHO 1 es falso, HECHO 2 es verdadero

Una regla FH relaciona hechos con hipótesis:

si HECHO 1 es falso y HECHO 2 es verdad (condición)

afirma HIPÓTESIS 1 con prioridad de 0,8

HH relaciona hipótesis con hipótesis:

si HIPÓTESIS 1 tiene prioridad de 0,4 a 1 y HIPÓTESIS2 tiene prioridad de 0,6 a 1

entonces HIPÓTESIS 3 tiene prioridad de 1.

Cuando más de una regla es aplicable a la misma hipótesis, la regla que se usa es la que tiene mayor índice de prioridad.

3.3.3. OPS5

Ops5 es un lenguaje de programación basado en reglas descendientes del lenguaje OPS diseñado para inteligencia artificial y aplicaciones de psicología.

Los datos Ops5 son vectores u objetos con pares de valores-atributos asociados.

Las reglas tienen la forma:

antecedente ----------> consecuente

donde el antecedente es una descripción parcial del elemento dado y el consecuente es una o más acciones que realizar si el antecedente se cumple.

3.4. Máquinas LISP

Los sistemas expertos se desarrollan y se entregan en muchas clases diferentes de equipos incluyendo grandes computadores, microcomputadores, computadores personales y estaciones de trabajo para inteligencia artificial especializadas o estaciones de trabajo de propósito general para ingeniería.

Como ya se ha comentado, a mediados de la década de los 70 se pusieron en marcha proyectos independientes en el Massachusetts Institute of Technology (MIT) y el Palo Alto Research Center, de la empresa Xerox, con el objeto de construir hardware específicamente adecuado a Lisp y a su entorno de desarrollo.

El proyecto de MIT (con fondos provenientes de una agencia del Departamento de Defensa) dio como resultado una máquina llamada CAR (tal como la función primitiva de Lisp) que, finalmente, sirvió de base a la primera máquina Lisp de tipo comercial, presentada por la empresa Symbolic a principios de la década de los ochenta.

Los sistemas expertos más grandes y más sofisticados tradicionalmente se desarrollaban en máquinas especializadas Lisp por la potencia y capacidad del equipo básico y además por los ambientes de software que corrían en ellas.

Todas las máquinas populares Lisp, tenían en común:

• Alta velocidad para el procesamiento de Lisp.

• Gran memoria física.

• Alta resolución, despliegue de pantalla por mapa binario.

• Uso de ratón para señalamiento.

• Enlace de comunicaciones.

• Soporte para ambientes de desarrollo de sistemas expertos potentes.

Como ya se ha dicho anteriormente, la desaparición de estas máquinas se produjo por su alto precio.

3.5. Grandes herramientas híbridas para Sistemas Expertos

Generalmente, las herramientas más poderosas para sistemas expertos son las grandes herramientas híbridas que combinan ambientes de desarrollo sofisticados con representaciones múltiples de conocimiento y paradigmas múltiples de inferencia. Esta integración, de varias facilidades, es muy útil porque posibilita que cualquier herramienta dada sea empleada con muchos y diferentes problemas, o problemas que involucran varias clases diferentes de representaciones del conocimiento y de paradigmas de inferencia.

Una herramienta híbrida posibilita el empleo de una sola herramienta para resolver problemas complejos grandes cuando diferentes partes del problema requieren diferentes clases de soporte.

Las herramientas híbridas grandes son usadas típicamente en estaciones de trabajo de inteligencia artificial, aunque se ejecutan en sistemas tradicionales. Hacen uso de gráficas de alta resolución de mapa binario, ventanería y ratón para proporcionar una interfaz con el usuario muy potente. Esta interfaz ayuda al desarrollador del sistema a construir el sistema precisa y rápidamente. Las mismas facilidades de interfaz pueden también individualizarse para brindar una interfaz potente para el usuario final.

3.6. ¿Cómo elegir una herramienta?

Elegir el ámbito adecuado del problema y la herramienta correcta para construir sistemas expertos, son dos de las decisiones que deben tomarse en el desarrollo de un sistema experto.

La mayoría de los sistemas expertos existentes fueron desarrollados con las herramientas elegidas, básicamente porque:

• El ingeniero del conocimiento estaba muy familiarizado con la herramienta.

• La herramienta era la única eficiente y disponible para ejecutarse en el hardware que se tenía.

• La herramienta era primeramente desarrollada y las aplicaciones eran hechas para probar la nueva herramienta.

La norma general es seleccionar inicialmente una herramienta y después evaluarla.

3.6.1. Selección de la herramienta

Hay seis preguntas básicas que se deben tener en cuenta cuando se elige una herramienta para un sistema experto:

1. ¿Proporciona la herramienta el conjunto de desarrollo con el poder y la sofisticación que se necesita?

2. ¿Las facilidades que soporta son adecuadas considerando el tiempo establecido para el desarrollo?

3. ¿Es fiable?

4. ¿Es mantenible?

5. ¿Tiene las características adecuadas a las necesidades del problema?

6. ¿Tiene las características adecuadas a las necesidades de la aplicación?

3.6.1.1. Restricciones de desarrollo

Los desarrollos de sistemas expertos requieren tiempo, dinero, personal y hardware. Todo ello influye a la hora de elegir una herramienta. En particular, estos factores influyen en la decisión acerca de qué tipo de herramienta seleccionar: un lenguaje de programación o un lenguaje específico para el desarrollo de sistemas expertos.

Los lenguajes de programación ofrecen más flexibilidad pero normalmente requieren del desarrollador, el diseño de la base de conocimientos y la implementación del motor de inferencia para el acceso al conocimiento.

Los desarrollos normales suelen ser largos con lenguajes de programación pero el resultado suele encajar más en las necesidades del dominio del problema.

Por otro lado, los lenguajes para ingeniería del conocimiento ofrecen menos flexibilidad pero más guías y mecanismos sobre cómo representar y acceder al sistema de conocimiento.

Con un lenguaje orientado al conocimiento, los desarrollos son más fáciles, más rápidos e incluso más económicos, pero no resultan tan efectivos ni tan eficientes como los sistemas escritos en un lenguaje de programación convencional.

3.6.1.2. Facilidades soportadas

Las facilidades soportadas por la herramienta apresuran los desarrollos y ahorran tiempo y dinero, por ello hay que elegir una herramienta con las facilidades adecuadas tales como editores, facilidades para entrada/salida, mecanismo de explicación, etc.

3.6.1.3. Fiabilidad

Los desarrollos se verán seriamente perjudicados si la herramienta no es fiable. Puede haber problemas por un incompleto testeo de la herramienta, una documentación obsoleta, unas especificaciones poco claras, etc.

Una herramienta será fiable si tiene una larga lista de usuarios y la precede una buena reputación. Cuando se considera una herramienta, se debe ver qué sistemas expertos han sido construidos con ella y qué opinan los desarrolladores, de la herramienta y de sus paquetes de utilidades.

No se debe construir un sistema experto con una herramienta que está todavía bajo desarrollo.

3.6.1.4. Mantenibilidad

Como dato a destacar en este apartado, simplemente comentar que lo mejor es elegir una herramienta donde el propio desarrollador no tenga que llevar a cabo el mantenimiento por si mismo durante el desarrollo del sistema experto.

3.6.1.5. Características precisas

Las tareas características del sistema influencian la elección de una herramienta.

Las características de un problema pueden dar lugar a diferentes tipos de soluciones que pueden sugerir diferentes tipos de herramientas. Así mismo, las características de la aplicación pueden sugerir características necesarias para el sistema en sí mismo, las cuales pueden sugerir una herramienta en particular.

3.6.2. Evaluación de la herramienta

Una vez seleccionada la herramienta hay que evaluarla. Para ello, se utiliza la herramienta para la resolución de un pequeño problema representativo del dominio de interés.

Este paso coincide con la necesidad de implementar un prototipo, para verificar que el ámbito del problema y el esquema de representación básicos, son estables.

Durante el testeo del prototipo, hay que prestar particular atención a la velocidad de ejecución del sistema. Si tarda horas, minutos o segundos o si se obtienen las respuestas deseadas, para el sistema desarrollado.

4. ADQUISICIÓN DEL CONOCIMIENTO

4.1. Introducción

La adquisición del conocimiento ha sido reconocida ampliamente como un arte y como un cuello de botella en la construcción de sistemas expertos.

Como puede suponerse fácilmente, los datos, informaciones y lo que es más importante, el conocimiento contenido en un sistema experto, puede proceder de muchas y variadas fuentes tales como libros, informes, bases de datos, estudio del caso, datos empíricos y experiencia personal. Sin embargo, la fuente dominante del conocimiento en los sistemas expertos actuales, es un experto en el dominio.

La tarea de adquisición del conocimiento a partir de expertos es en la actualidad una labor de artesanía hecha para cada caso. Para realizar esta labor hay que entrevistar a los expertos y/o analizar sus protocolos; proponer y estudiar los problemas típicos, ejemplares y de prueba; identificar los conceptos, adquirir y formular, además de reglas y/o heurísticas, los distintos niveles: estructurante, conceptual y cognoscitivo.

Un intento de construir una alternativa al actualmente azaroso y artesanal proceso de construir y mejorar las bases de conocimiento, es mecanizar al máximo la labor que actualmente realizan los ingenieros del conocimiento. El desarrollo de estas metodologías estándares puede ser el punto de partida para conseguir, por una parte, la automatización de la adquisición del conocimiento y, por otra, la construcción de bases de conocimiento reusables.

Pero como pasará algún tiempo antes de que los métodos automáticos puedan incluso empezar a reemplazar a los ingenieros del conocimiento, es necesario analizar cómo estos adquieren el conocimiento a partir del experto.

4.2. Relación entre el experto y el ingeniero del conocimiento

El ingeniero del conocimiento juega un papel crítico en la construcción de un sistema experto. Aunque es el conocimiento del experto el que se está modelando, es el ingeniero del conocimiento quien debe realmente construir el sistema.

En este proceso, el ingeniero del conocimiento actúa como un intermediario que espera conducir dicho proceso, llevando el conocimiento del experto al sistema. Esto es en particular crítico, porque el conocimiento del ingeniero del conocimiento, más que el del experto, es el reflejado realmente en el sistema experto.

Como consecuencia, el ingeniero del conocimiento debe ser muy cuidadoso para reflejar el conocimiento del experto; así como un traductor de idiomas debe buscar el sentido que quiere darle el autor a sus palabras.

El primer paso en el proceso de adquisición del conocimiento es la Comprensión del Dominio. Esto es, un período de familiarización general.

Este paso comienza con una reunión informal entre el ingeniero del conocimiento y el experto.

Los propósitos de esta reunión son:

• Comenzar estableciendo una armonía entre el ingeniero del conocimiento y el experto. El aumento de la armonía se debe basar en el mutuo respeto. El ingeniero del conocimiento respeta al experto por su comprobada maestría del dominio y el experto respeta al ingeniero del conocimiento por su interés en el dominio y voluntad por aprender. La relación, con frecuencia, sigue la analogía instituida entre el maestro y el aprendiz. Si dicha armonía no se puede establecer, se debe conseguir un nuevo experto o un nuevo ingeniero del conocimiento o se debe abandonar el proyecto.

• Proporcionar al ingeniero del conocimiento un panorama de muy alto nivel del dominio.

• Poner al corriente al experto con conceptos de sistemas expertos y con respecto a los papeles de un experto y de un ingeniero del conocimiento. El ingeniero del conocimiento debe mostrarle al experto un ejemplo de un sistema similar y que funcione, si es posible.

El ingeniero del conocimiento debe ser consciente de la necesidad de una relación trabajo-personal con el experto.

Esto es difícil en algunas ocasiones porque el ingeniero del conocimiento, por lo menos inicialmente, debe tomar una posición secundaria respecto al experto y debe también tolerar las peculiaridades personales de éste.

Durante el periodo restante de familiarización o de comprensión del dominio que hablábamos antes, el ingeniero del conocimiento trabaja en alto grado a solas con los materiales básicos, tales como documentos de referencia, material de entrenamiento y videocintas, intentando desarrollar una visión global del dominio, mientras hace un intento consciente de evitar detalles.

En esta parte, el ingeniero del conocimiento intenta aprender acerca del dominio en la misma forma que lo hizo el experto.

De este estudio resultan varios elementos importantes:

• Una descripción general del problema.

• Una bibliografía con referencias importantes.

• Un glosario que describe el lenguaje del dominio. Esto incluye los términos, siglas y símbolos que son característicos del dominio. El glosario es particularmente importante, ya que el conocimiento del vocabulario es uno de los tipos más fundamentales de conocimiento en cualquier dominio.

Siempre que sea posible, el ingeniero del conocimiento debe usar el lenguaje corriente del dominio cuando trabaje con el experto.

El siguiente paso en el proceso consiste en la Identificación de los Problemas de Muestra que van a ser usados en la implementación inicial.

De una lista de problemas genéricos, que sirven para circunscribir el espacio global del problema, se seleccionan varios problemas específicos, los cuales son identificados por el ingeniero del conocimiento y por el experto trabajando conjuntamente. Estos deben ser característicos de una clase de problemas, importantes y bien entendidos.

A continuación de este proceso de selección, comienzan en paralelo las tres actividades más críticas. Formulación Conceptual, Implementación de los Problemas de Ejemplo y el Desarrollo de la Representación de Conocimientos.

La implementación de los problemas de muestra es conducida por el ingeniero del conocimiento, quien entrevista al experto, en relación con la actividad de solución de problemas.

La estructura del conocimiento, que se empleará para representar los conceptos de desarrollo y el conocimiento de la solución de problemas, tomará forma según avancen los otros procesos.

Cuatro dificultades básicas se pueden presentar durante estas actividades:

- El experto puede no tener el conocimiento requerido en algún área.
- El experto puede no ser consciente del conocimiento requerido.
- El experto puede no estar capacitado para comunicar el conocimiento al ingeniero del conocimiento.
- El experto puede no estar en capacidad de estructurar el conocimiento para introducirlo en la base de conocimientos.

Cuanto mas hábil sea el ingeniero del conocimiento, tanto más el sistema final reflejará las heurísticas reales y los procedimientos operativos del experto.

4.3. Técnicas de adquisición del conocimiento

Cuando los expertos resuelven problemas en su área de experiencia, reconocen nuevas situaciones como instancias de cosas con las cuales ya están familiarizados. Sin embargo, cuando los expertos se enfrentan con situaciones nuevas, se comportan como unos simples novicios inteligentes y tienden a aplicar principios generales y pasos deductivos, que proporcionan enlaces causales entre varias etapas de una secuencia de soluciones de problemas.

Estas diferencias en las estrategias para resolver problemas, sugieren algunas técnicas para la adquisición del conocimiento.

Una posibilidad es presentar situaciones nuevas, tal vez sugeridas por otros expertos, y anotar el proceso que sigue el experto para resolver el problema. Es lo que se denomina método del protocolo.

La manera tradicional de adquisición del conocimiento a partir de expertos mediante el diálogo con los ingenieros del conocimiento, presenta dos graves inconvenientes:

• El primero, es que conduce a un proceso lento, laborioso y costoso, pues, en particular, el ingeniero del conocimiento tiene que familiarizarse con el lenguaje y los conceptos básicos que maneja el experto.

• El segundo, estriba en la dificultad que manifiesta el experto para hacer explícito su conocimiento en uno de los formalismos de representación existentes. Sin embargo, se ha observado que en muchos campos existen expertos que son capaces de solucionar problemas, pero incapaces de explicar detalladamente los procesos mentales que les han llevado a alcanzar tales soluciones.

En estos casos, puede resultar útil que el sistema "induzca" procedimientos de soluciones generales a partir de soluciones dadas por los expertos a casos particulares. En este caso, el experto es quien refina su propio conocimiento y sus reglas mediante la facilidad que tienen los expertos para enseñar a través de inducción.

Las condiciones necesarias para que pueda darse este conocimiento por inducción son: que el dominio del conocimiento pueda caracterizarse a través de ejemplos de soluciones dadas por el experto humano; que el experto humano pueda dar esos ejemplos en función de los atributos o los descriptores que llevan a la solución; y, finalmente, que los ejemplos sean razonablemente complejos.

La idea de inducción del conocimiento es generalizable en el sentido que, en muchos dominios de las experiencias, el conocimiento se va acumulando obteniéndose nuevos conocimientos o perfeccionándose los ya acumulados mediante el diseño y análisis de los resultados obtenidos, a partir de experimentos controlados.

Este método, pone de manifiesto los procedimientos y estructuras de conocimiento implícitas en la mente de los expertos.

Por ejemplo, el experto hará introspección, mientras resuelve un problema proporcionado por el ingeniero del conocimiento y habla en voz alta sobre cómo lo está resolviendo. El ingeniero del conocimiento lo orienta siempre que lo crea conveniente, planteando cuestiones relevantes, sugiriendo posibles razones, e "hipotetizando" conceptos y reglas.

Los psicólogos han estudiado a los expertos y sus técnicas de resolver problemas, usando métodos intuitivos y de observación o experimentales, para medir las prestaciones y descubrir la experiencia. El método experimental, confía en observar al experto resolviendo problemas realistas

y siendo cuidadoso en no decir o hacer algo que pueda influenciar el enfoque del experto para resolver el problema.

Los métodos de observación son a veces seguidos de una fase de refinamiento, en la cual, los expertos comentan sobre los modelos preliminares desarrollados para descubrir su comportamiento.

Los investigadores que aplican el método de observación, no interrumpen al experto con cuestiones o comentarios durante la solución del problema, antes bien, analizan una transcripción de la sesión después de haberla finalizado, posiblemente, con la ayuda del experto. Este método, denominado "análisis de protocolo", ha sido usado por los investigadores de inteligencia artificial para estudiar la solución de problemas por los no expertos. En este caso, el sujeto habla mientras resuelve un problema sencillo, las verbalizaciones se graban y el proceso de resolución subyacente se infiere de la traza resultante.

Con el método de observación se resuelve un problema mientras se describe en voz alta lo que se está haciendo. Aunque el hecho de pensar en voz alta puede alterar un poco la técnica del experto, el problema surge en las enormes lagunas que ocupan con frecuencia la descripción del proceso, especialmente cuando el experto incrementa el conocimiento compilado.

Por otro lado, el método intuitivo tiene diferentes tipos de problemas. Ya que el experto usa la introspección, tiene dificultades de exponer las técnicas reales usadas en resolver los problemas. El conocimiento está tan bien comprobado a través de la experiencia y una práctica continuada, que el experto lo accede y manipula sin pensarlo.

4.4. Enfoques para la extracción y verificación del conocimiento

Las formas más habituales usadas para extraer el conocimiento son las siguientes:

1. Observación al lado. En este caso, el ingeniero del conocimiento observa al experto resolviendo problemas reales sobre el trabajo, antes de "inventar" problemas realistas de laboratorio. Aquí no se interfiere, sino que más bien se actúa como un observador pasivo. Esta técnica, no será práctica ni útil en algunos dominios, habitualmente por consideraciones de tiempo o privacidad.

2. Disección del problema. Aquí, el ingeniero del conocimiento escoge un conjunto de problemas representativos e informalmente los discute con el experto. La meta es determinar de qué manera el experto organiza el conocimiento acerca de cada problema, representa los conceptos e hipótesis, y maneja el conocimiento y los datos inconsistentes, inexactos o imprecisos relativos al problema. Durante esta discusión, el experto puede introducir nuevos términos, conceptos y relaciones. Cuando esto sucede, el ingeniero del conocimiento le pide al experto que defina esas nuevas construcciones y las relacione con los conceptos existentes en la base de conocimientos desarrollada.

3. Descripción del problema. En este caso, el ingeniero del conocimiento le pide al experto que describa un problema típico para cada categoría importante de respuestas que pueden surgir. Esto ayuda al ingeniero del conocimiento a definir problemas típicos para cada categoría de respuestas. Este enfoque trabaja particularmente bien para problemas de diagnóstico médico y mecánico.

4. Análisis del problema. En este enfoque, el ingeniero del conocimiento le pide al experto que resuelva una serie de problemas, probando el razonamiento del experto a medida que se resuelven los problemas. Mientras el experto resuelve cada problema, el ingeniero del conocimiento proporciona cualquier información adicional o dato requerido por el experto. El experto debe resolver problemas realistas, describiendo el proceso de solución en voz alta y dando tantos pasos intermedios como sea posible. Una vez que el ingeniero del conocimiento ha formulado algunas reglas especializadas relativas a problemas particulares, puede revisar las reglas para hacerlas tan generales como sea posible. Para este proceso puede ser necesaria la asistencia del experto.

5. Refinamiento del sistema. El experto le da al ingeniero del conocimiento problemas para resolver, clasificándolos de muy fáciles a francamente difíciles. Antes de que el sistema experto sea operativo, el ingeniero del conocimiento los resuelve en el papel usando los conceptos, formalismos y reglas adquiridas del experto. Esto proporciona una comprobación rápida de la consistencia y completitud del conocimiento que está siendo extraído del experto.

6. Examen del sistema. El experto examina y critica cada regla en el prototipo y evalúa las estrategias de control empleadas para seleccionar las reglas. Esto incluye verificar la exactitud de cada regla y establecer una justificación para cada una, de modo que el sistema pueda posteriormente usarla para explicar su operación. El experto debería comparar las estrategias de control en el prototipo con su método de manejar problemas en el dominio.

7. Validación del sistema. El ingeniero del conocimiento presenta los casos resueltos por el experto y el sistema prototipo a otros expertos. Esto proporciona una manera de comparar estrategias de diferentes expertos y encontrar puntos esenciales de desacuerdo.

4.5. Herramientas para adquisición de conocimientos

Debido a que la adquisición de conocimientos es una tarea difícil y que consume tiempo, sería extremadamente útil reemplazar al ingeniero del conocimiento por una herramienta que le permitiera al experto construir el sistema directamente.

Tal herramienta haría posible que un experto, que no estuviera familiarizado con conceptos de inteligencia artificial, pudiera construir un sistema de forma rápida y sencilla. Pero, aunque el desarrollo de las herramientas para la adquisición de conocimientos parece muy positivo,

también queda aún por ver la efectividad de las herramientas cuando se aplican a dominios grandes y variados, o la eficacia de los sistemas resultantes, con relación a los sistemas construidos por los ingenieros del conocimiento humanos.

4.5.1. TEIRESIAS

Teiresias fue un intento temprano de construir una herramienta para la adquisición de conocimientos. Tuvo el propósito de ayudar a un experto a mantener y a ampliar la base de conocimientos de Mycin.

Interactúa con el experto empleando un subconjunto restringido del inglés, para agregar nuevas reglas y corregir las existentes. Una de las formas primarias en que un experto emplea a Teiresias, es ejecutando Mycin y usando las facilidades de explicación para observar cualquier error, que luego se corrige mediante el empleo de Teiresias.

4.5.2. ROGET

Roget es la primera herramienta para construir un sistema experto original a través de la interacción directa del experto.

Roget ayuda al experto desde las etapas iniciales del diseño, hasta la etapa de construcción. La ayuda proporcionada incluye el desarrollo de estructuras conceptuales del sistema, estrategias de inferencia, identificación de factores críticos, etc.

Sugiere, también, una visión razonable en cuanto al sistema y recomienda las herramientas apropiadas.

Como primer paso en este proceso, Roget, establece la estructura conceptual. Para establecer esta estructura, le proporciona al experto ejemplos de sistemas expertos existentes y le pide al experto identificar el que más se parezca al sistema que se desea desarrollar.

Después de que se termine la definición completa del diseño, comienza la adquisición de conocimientos detallada sobre problemas específicos. El conocimiento se obtiene, primero, en profundidad, procediendo a través de varios niveles de recolección de conocimiento, denominados categorías de evidencia.

A medida que los problemas de ejemplo se completan, los fragmentos del sistema resultante son realimentados al experto (empleando un subconjunto del inglés) para la revisión y el repaso. El conocimiento detallado se revisa luego (principalmente mediante poda de la base de conocimientos) y el resultado se presenta al experto para su verificación final.

Después de esta verificación, Roget traduce los conocimientos desde su representación interna a las reglas ejecutables reales. El resultado de

este proceso es un prototipo de un sistema experto apropiado en cuanto a ejecución y evaluación, después de un número suficiente de sesiones.

4.5.3. KADS

Kads es una metodología que fue desarrollada mediante un proyecto de la Comunidad Europea.

Esta metodología, permite modelizar el conocimiento del sistema, mejorando la eficacia de esta actividad crítica en todo desarrollo de sistemas expertos.

Para realizar la adquisición de conocimientos de forma asistida mediante Kads, se utiliza un conjunto de modelos que describen cómo realizar una tarea determinada, a partir de librerías ya predefinidas y modelos adaptables, ya creados en base a experiencias anteriores, siendo posible rediseñar modelos de manera fácil. Estos modelos incluyen tres niveles de descripción:

• Nivel de dominio: contiene objetos, jerarquías, definiciones y grafos de relación entre clases. Este nivel está basado fundamentalmente en una descripción orientada a objetos que determina el formalismo a aplicar en el resto de los niveles

• Nivel de inferencia: en este nivel se describe el flujo de datos o de conocimientos entre los distintos pasos de razonamiento. Este nivel permite un punto de vista funcional de la aplicación.

• Nivel de tarea: a partir del cual se dividen las distintas tareas a realizar en subtareas y en sus correspondientes agendas. Mediante este nivel, es posible obtener las estructuras de control que organicen las actividades hacia la solución del problema.

4.5.4. KAS

Es un sistema de adquisición de conocimiento para Prospector. Un programa de consulta desarrollado para problemas de diagnóstico en explotación de minerales.

Kas tiene la misma relación con Prospector como Emycin la tiene con Mycin.

Kas no es un lenguaje de programación.

5. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

5.1. Herramientas utilizadas

En este apartado se explicará el motivo por el cual se escogió Visual Basic para el desarrollo del sistema y no un lenguaje de propósito general como C o Fortran o algún lenguaje específico como Lisp o Prolog.

También se hablará de la base de datos utilizada para almacenar la base de conocimientos del sistema.

5.1.1. El porqué de la elección de Visual Basic

Como ya se ha visto, para el desarrollo de un sistema experto podemos utilizar lenguajes específicos como Lisp o Prolog o lenguajes de propósito general como C o Fortran, pero, ¿por qué se eligió Visual Basic para el desarrollo de este proyecto?.

Existen varias razones. La razón principal por la que se decidió utilizar Visual Basic y no un lenguaje específico o una herramienta para el desarrollo de sistemas expertos que incorporase el resolvedor de problemas (o motor de inferencia) es precisamente el reto que suponía el hecho de partir de cero e implementar el sistema en su totalidad, incluyendo el resolvedor de problemas.

Implementar el resolvedor de problemas ha resultado, con mucho, la parte más complicada y extensa de este proyecto. También ha resultado ser la más ambiciosa e interesante desde el punto de vista técnico, tanto por la programación del motor de inferencia como por el diseño del mismo.

Se escogió, entre otras, la herramienta Visual Basic porque permite la creación de interfaces gráficas y porque es un lenguaje de programación dirigido por eventos. Ambas particularidades parecían encajar bastante bien para el desarrollo del sistema experto.

5.1.2. Utilización de Microsoft Access como base de datos

Microsoft Access permite crear ficheros de bases de datos relacionales que pueden ser fácilmente gestionadas por una interfaz gráfica simple. Además, estas bases de datos pueden ser consultadas por otros programas. Microsoft Visual Basic proporciona componentes específicos para tratar con bases de datos Access.

Se necesitaba una base de datos sencilla en la cual almacenar los datos de la base de conocimientos sin mayores pretensiones y la herramienta Microsoft Visual Basic daba todas las facilidades para interactuar con ella, al ser ambos productos de Microsoft. Es por todo ello que se optó por esta base de datos entre otras posibilidades.

5.2. La interfaz gráfica

La interfaz gráfica del sistema está dividida en tres secciones.

En la primera sección podemos distinguir el título de la ventana y opciones de menú con las distintas acciones que el usuario puede realizar a modo de menús desplegables.

En otra de las secciones se pueden apreciar los diagnósticos de la base de conocimientos representados a modo de árbol con la jerarquía de dependencia entre unos y otros.

Y en la última de las secciones se muestran las manifestaciones de la misma manera que los diagnósticos en una estructura de árbol jerárquico en el que se aprecia la dependencia entre unas manifestaciones y otras.

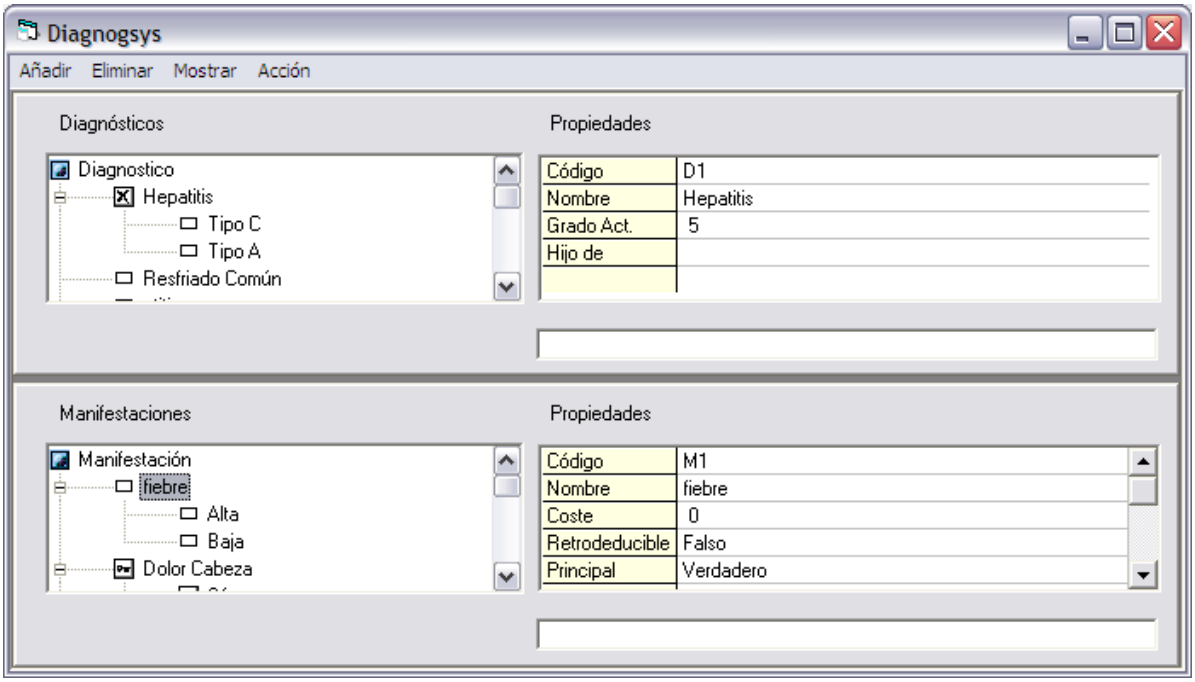


Figura 5.1. Pantalla principal

En la figura 5.1 podemos distinguir la tres secciones en que se divide la pantalla principal y en la parte derecha de la misma las propiedades del diagnóstico y manifestación seleccionados en el árbol correspondiente.

Las propiedades de los diagnósticos se muestran en la parte derecha del árbol de diagnósticos. Como se puede apreciar en la figura 5.1 el diagnóstico “Hepatitis” se encuentra seleccionado y se muestran los valores de las propiedades Código, Nombre, Grado Act. e Hijo de. Estos valores pueden ser modificados por el usuario, a excepción del Código, cuyo valor lo asigna automáticamente el sistema.

De la misma manera que con los diagnósticos, las propiedades de la manifestación seleccionada se muestran en la parte derecha del árbol de manifestaciones. Se observa que la manifestación “Fiebre” se encuentra seleccionada y se muestran los valores de las propiedades Código, Nombre, Coste y Retrodeducible, además de otras más. También sus valores son modificables por el usuario, a excepción de la propiedad Código.

El significado de las propiedades y sus posibles valores de los diagnósticos y de las manifestaciones se explicarán con detalle en secciones sucesivas de este documento.

En ambos árboles de diagnósticos y de manifestaciones se pueden apreciar diferentes iconos acompañando al nombre del diagnóstico o de la manifestación.

Así, el icono  representa el nodo raíz del árbol.

El icono  representa una manifestación desactivada.

El icono  representa una manifestación activada.

El icono  representa un diagnóstico activado.

El icono  representa un diagnóstico desactivado.

Más adelante se detallará la manera en la que el usuario puede activar o desactivar un diagnóstico o manifestación para limitar la base de conocimientos cuando se tiene constancia de que un determinado diagnóstico o manifestación que, estando presente en la base de conocimientos, no se va a reproducir en el escenario inicial del que parte el resolvedor de problemas.

5.3. Funcionalidad del Sistema

El sistema diseñado tiene los siguientes componentes principales:

- Un árbol gráfico de diagnósticos, donde se lleva a cabo la jerarquización de los distintos diagnósticos que se pueden presentar en el sistema.
- Un árbol gráfico de manifestaciones, donde se lleva a cabo la jerarquización de las distintas manifestaciones que se pueden presentar en el sistema y que están relacionadas con los diagnósticos.
- Tablas de propiedades asociadas a cada tipo de objeto que podemos encontrar en la base de conocimientos.
- El resolvedor de problemas o motor de inferencia que permitirá obtener un diagnóstico gracias a las manifestaciones que se presenten durante el problema a resolver y al algoritmo encargado de encauzar la solución final.

Se trata de diseñar un sistema en el que, dadas unas manifestaciones ocurridas en el mundo real, determine el/los diagnóstico/s más adecuado/s que se ajuste/n a esas manifestaciones, para, finalmente, escoger el diagnóstico final que será el más adecuado de todos.

El sistema consta de un conjunto de diagnósticos, de otro conjunto de manifestaciones y de las relaciones entre estos dos conjuntos. Estas relaciones determinan lo que tienen que ver unas determinadas manifestaciones con unos determinados diagnósticos (nada, mucho, poco, algo,...).

Tanto el conjunto de diagnósticos, como el conjunto de manifestaciones y las relaciones entre ellos, que componen el sistema, los crea el usuario gracias a la interfaz, a modo de ventanas con “árboles”.

Mediante las relaciones entre diagnósticos y manifestaciones se establecen vínculos que nos van a permitir conocer el grado de participación de una determinada manifestación con un determinado diagnóstico en el proceso de lógica que nos llevará a obtener un solo diagnóstico de entre todos los posibles, como solución al problema.

Una vez que tenemos el conjunto de manifestaciones y diagnósticos y sus relaciones, el usuario puede activar aquellas manifestaciones que crea oportunas y que van a permitir potenciar la candidatura de unos posibles diagnósticos y despotenciar la de otros, así como desactivar aquellas manifestaciones que le constan no se presentan en el problema a resolver.

Algunas de las manifestaciones son positivas y otras negativas. Las manifestaciones positivas sugieren ciertos diagnósticos, por el contrario, las manifestaciones negativas desaconsejan ciertos diagnósticos.

Una manifestación sugiere (mediante un determinado valor) un diagnóstico. Esto hace que, si la manifestación está presente en el problema a resolver y, por tanto, está activa, el **grado de activación** del diagnóstico relacionado con ella aumente frente a otros.

A partir de las manifestaciones activas positivas calculamos el grado de activación de los diagnósticos que sugieren.

Por lo tanto, una de las variables a tener en cuenta en el cálculo del grado de activación de un diagnóstico, es el valor de la propiedad sugerencia de la manifestación activa positiva relacionada con el diagnóstico.

El grado de activación de un diagnóstico determina la idoneidad de ese diagnóstico para ser la solución al problema, de tal forma que la solución del problema será aquel diagnóstico con mayor grado de activación cuando se acabe el proceso de búsqueda o “razonamiento”.

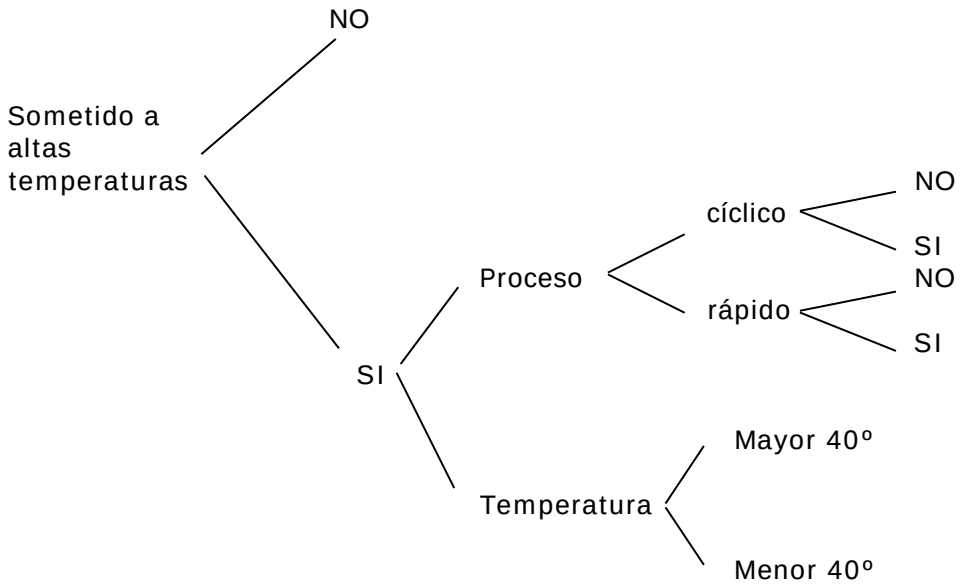
Al igual que las manifestaciones activas positivas aumentan el grado de activación de los diagnósticos mediante el valor de la propiedad

sugerencia, las manifestaciones activas negativas disminuyen el grado de activación de los diagnósticos mediante el valor de la propiedad **frecuencia**. He aquí que las manifestaciones activas positivas sugieren a un determinado diagnóstico, y, por el contrario, las manifestaciones activas negativas, desaconsejan a los diagnósticos con que se relacionan.

Una manifestación es positiva o negativa dependiendo del valor de la propiedad **positiva** S/N.

Se ha mencionado que las manifestaciones (al igual que los diagnósticos) están presentes en el sistema formando una estructura de árbol. De esta forma hay manifestaciones padre, manifestaciones hija y manifestaciones hermanas. Por lo tanto, una determinada manifestación puede tener descendientes que se activarán si se activa la manifestación padre. Estas manifestaciones descendientes activadas, participarán, a modo de cascada, en el grado de activación de los diagnósticos con que se relacionan.

Veamos un ejemplo de estructura en árbol para entender el concepto de todas estas propiedades y algunas más que se detallan:



- Se dice que una manifestación es **retrodeducible** cuando conociendo la ocurrencia de dicha manifestación se puede conocer la ocurrencia de la manifestación padre.

- La manifestación “Sometida a altas temperaturas/si/proceso/cíclico/si” es retrodeducible ya que de dicha manifestación podemos deducir que ha sido sometido a altas temperaturas.

- Sin embargo, la manifestación “Sometida a altas temperaturas/si/proceso/cíclico/no” no es retrodeducible ya que el saber

que no ha sufrido procesos cíclicos no nos dice nada acerca de si ha estado sometido a altas temperaturas.

- Se dice que una manifestación es **excluyente** cuando la afirmación de esa manifestación implica la negación de sus hermanas.

- La manifestación "Sometida a altas temperaturas/no" es excluyente de la manifestación "Sometida a altas temperaturas/si", y viceversa.

Cuando se activa una manifestación excluyente se niegan sus hermanas y todas las descendientes de estas.

Cuando el sistema comience el proceso de lógica que permita obtener la solución al problema, determinará qué camino seguir en el árbol de manifestaciones, realizando **preguntas** al usuario.

Estas preguntas irán asociadas a las manifestaciones.

Existen manifestaciones **primarias** y manifestaciones **secundarias**.

Las manifestaciones primarias son aquellas que tienen algún valor en su propiedad pregunta.

Las manifestaciones secundarias son aquellas que no tienen ningún valor en su propiedad pregunta.

Cuando el sistema, en un proceso de deducción, necesita realizar una pregunta al usuario para conocer la ocurrencia de una determinada manifestación secundaria, observa si la manifestación padre es una manifestación principal. Si es una manifestación principal o primaria, el sistema preguntará al usuario la pregunta que la manifestación principal tenga. Si no es una manifestación principal, el sistema tomará como pregunta todo el camino que va desde la manifestación raíz hasta la manifestación a preguntar.

Una manifestación principal solo puede tener manifestaciones secundarias como hijas.

Una manifestación secundaria puede tener manifestaciones principales tanto como hijas como padres.

Al asertar o activar una determinada manifestación se activan sus ascendientes.

Cuando una determinada manifestación no se produce en nuestro problema se retracta o desactiva y, por tanto, todos sus descendientes.

Una manifestación puede tener tres estados:

- ACTIVADA: manifestación que está presente en el problema.

- DESACTIVADA: manifestación que no está en el problema.

- NEUTRO: estado de partida. Se desconoce si la manifestación está o no presente en el problema. Estas son las únicas que el sistema preguntará al usuario.

5.4. Adquisición del conocimiento

En un principio se creyó que no sería posible un sistema experto sin la existencia y colaboración de un experto humano. Sin embargo, hoy se sabe, como luego se podrá demostrar, que son posibles sistemas expertos sin la colaboración directa de un experto en el tema de estudio.

De todas formas, la mayoría de los sistemas expertos necesitan la colaboración de los expertos humanos y de los ingenieros del conocimiento.

En el sistema experto desarrollado, no ha intervenido un experto como tal. No se han llevado a cabo entrevistas o encuentros con un determinado experto en el ámbito del problema para adquirir el conocimiento que este posee, sino que, toda la información contenida en la base de conocimiento del sistema proviene de:

- La lectura de libros y revistas especializadas en medicina.

- Consultas puntuales a un profesional de la medicina que, por motivos de parentesco lejano, estaba predispuesto a compartir su conocimiento de manera distendida y en un ámbito exclusivamente familiar.

- Consultas en diversos portales web, como, por ejemplo, Wikidoc, sobre la relación que puede existir entre una determinada manifestación y uno o varios diagnósticos.

Así pues, podemos ver que en la elaboración de la base de conocimientos de este proyecto no se ha empleado una metodología estándar como tal.

5.4.1. Problemas en la adquisición del conocimiento

Aunque la adquisición del conocimiento fue sencilla, porque no hubo que establecer una difícil relación con un experto, presentó algunos problemas.

En principio, cuando se empezó con la búsqueda de información se decidió comenzar con la lectura de libros y revistas científicas. Los libros y las revistas generalmente eran muy especializados y se hablaba de conceptos que se desconocían como mialgias, artralgias, neuraminidasa, exitus letalis, etc. Así que se optó por restringir la búsqueda de información a posibles manifestaciones que puede declarar un enfermo en la consulta de su médico de cabecera, las más comunes, y sus posibles diagnósticos. Para ello se hizo uso de las posibilidades que hoy en día nos ofrece la Red,

realizando búsquedas en Wikipedia o en Wikidoc, enciclopedia virtual especializada en temas de medicina.

5.5. Diseño de la base de conocimientos

Se considera una base de conocimientos como un conjunto de objetos que tienen una relación jerárquica en el conocimiento de un área determinada.

Por ejemplo, se puede tener fiebre, pero dicha fiebre puede ser alta o muy alta, y a su vez, puede venir acompañada de malestar general o no.

Como se puede ver, estamos estableciendo relaciones jerárquicas entre manifestaciones como fiebre, que a su vez puede ser alta o muy alta, y que, a su vez, se relacionan con diagnósticos como gripe común o neumonía vírica, por poner un ejemplo.

Una base de conocimientos está compuesta de objetos.

Un OBJETO es un ente que posee unas determinadas propiedades.

En el sistema experto que hemos diseñado nos encontramos tres tipos de objetos: diagnósticos, manifestaciones y las relaciones entre ellos.

Las PROPIEDADES son atributos de un objeto y contienen toda la información que la base de conocimientos dispone acerca de un objeto en particular.

Por ejemplo, basándonos en el caso anterior, es interesante saber cuáles pueden ser las manifestaciones más habituales que se pueden presentar cuando tenemos gripe, cómo podemos relacionarlas con los posibles diagnósticos o con qué frecuencia aparecen esos síntomas o manifestaciones ante un diagnóstico u otro.

Todo esto, se va a almacenar en las propiedades de un determinado tipo de objeto de la base de conocimientos.

Una propiedad va a estar compuesta de un nombre y un valor.

El conocimiento almacenado dentro de una base de conocimientos, debe provenir siempre de un experto, para que dicho conocimiento sirva de guía y experiencia a otras personas no expertas, que serán los usuarios finales del sistema.

La base de conocimientos que se ha diseñado es pasiva, es decir, por sí sola no puede funcionar, es el resolvedor de problemas el que hará funcionar a la base de conocimientos.

5.6. Implementación de la base de conocimientos

En este apartado se describirá la manera en la que el resolvedor de problemas utiliza un determinado algoritmo para la búsqueda de soluciones.

Se pretende ir mostrando paso a paso cómo se ha implementado el sistema.

5.6.1. Implementación de los datos iniciales

La información existente en la base de conocimientos está organizada de forma jerárquica en un árbol causal de manifestaciones y en un árbol de diagnósticos que están relacionados con esas manifestaciones por medio de las relaciones existentes en la base de conocimientos.

Así, podemos tener la manifestación “Dolor de cabeza → Agudo”, el diagnóstico “Migraña”, y la relación “Dolor de cabeza → Agudo <-> Migraña”, que relaciona la manifestación con el diagnóstico, por ejemplo.

5.7. Proceso de razonamiento

5.7.1. Introducción

El proceso de razonamiento se basa en un modelo atencional mediante el cual el sistema focaliza su atención en diagnósticos y manifestaciones para poder resolver en primera instancia el problema:

> Dadas unas manifestaciones ocurridas en el mundo real, ¿cuáles son los diagnósticos más adecuados para explicar su ocurrencia?.

Focalizando de esta forma la atención hacia un determinado número de diagnósticos, una vez focalizada se resuelve el siguiente subproblema:

> Dados esos diagnósticos que son los más probables, ¿qué manifestaciones necesito conocer, para discernir cuál de los diagnósticos elijo?.

Veamos durante un momento cual es el proceso de resolución habitual de una persona. Tomamos el mayor número de hechos del mundo real, los analizamos y obtenemos unas hipótesis, buscamos nuevos datos del mundo real que nos confirmen o nieguen con más fuerza alguna de las hipótesis elegidas y vuelta a comenzar, es decir, fijamos nuestra atención primero en determinados diagnósticos y posteriormente realizamos una previsión acerca de la estrategia que encauzará el problema, prosiguiendo a partir de datos del mundo real a elegir nuevas hipótesis hasta resolver el problema. Esta manera recurrente es muy habitual en la resolución de problemas y es similar a la seguida por el sistema.

El comienzo del proceso de razonamiento se realiza por la introducción previa de manifestaciones que puede realizar el usuario a la hora de consultar el sistema. Esta introducción se realiza directamente en el árbol de manifestaciones. De igual manera se procede con los diagnósticos y se deben establecer las relaciones existentes entre los diagnósticos y manifestaciones presentes en la base de hechos.

Tras este proceso, el sistema comienza el proceso de razonamiento preguntando por las manifestaciones primarias, que son aquellas que tienen valor en la propiedad pregunta.

De esta forma conseguimos un conjunto de manifestaciones activas, algunas de ellas positivas y otras negativas, que aconsejan o desaconsejan los diagnósticos con los que se relacionan.

A partir de las manifestaciones activas positivas y en conjunción con los conjuntos difusos que las enlaza con los diagnósticos se obtiene la activación de los diagnósticos con los que se relacionan aumentando el grado de activación de los mismos, algo fundamental a la hora de encontrar el diagnóstico que ponga fin a la resolución del problema.

La activación de los diagnósticos se realiza mediante una pregunta realizada al usuario por cada manifestación activa y relacionada con ellos, calculándose, a partir de los datos del enlace, el grado de activación. De esta forma se consigue "excitar" determinadas representaciones de la memoria que dirigirán a partir de ahora el proceso de razonamiento.

De lo que se trata, en el mecanismo de razonamiento, es intentar que uno de los diagnósticos posibles tenga muchísima más atención prestada que al resto, esto es, mayor grado de activación que el resto.

5.7.2. Lógica difusa

La lógica difusa es una teoría que se ha implantado en el campo científico-técnico y que en definitiva nos resulta realmente útil si nos interesa que un determinado dispositivo (máquina, programa, aplicación, …) "piense" tal y como lo haría la mente humana. Esta lógica se basa fundamentalmente en crear una relación matemática entre un elemento y un determinado conjunto difuso con el fin de que una computadora sea capaz de realizar una valoración similar a como lo hacemos nosotros.

Supongamos que estamos hablando de la mediana edad. Al escuchar este término mentalmente lo asociamos a un determinado tipo de imágenes y personas que a una máquina le sería imposible realizar porque es incapaz de razonar y comprender un aspecto abstracto o impreciso.

Para conseguirlo, la lógica difusa utiliza una función de pertenencia [0,1] entre un elemento (en nuestro caso serán los años) y un determinado conjunto que a priori será confuso para el computador (para nosotros la mediana edad).

Partimos de la base que la mediana edad son los 45 años. En ese caso, la función de pertenencia entre los años y la mediana edad será máxima y valdrá 1. Sin embargo, no podemos descartar a las personas de 35 ó 55 años como mediana edad. Por otro lado, los menores de 35 años y los mayores de 55 tampoco se pueden considerar radicalmente que no pertenecen a la mediana edad aunque el grado de pertenencia a la misma será mucho menor y cada vez más cercano a cero.

Con esta teoría conseguimos que esa relación matemática que hemos obtenido, gracias a la función de pertenencia, forme la base para que una máquina sea capaz de interpretar si un elemento pertenece o no a un conjunto difuso y, lo que es más importante, que pueda evaluar si ese grado de pertenencia es elevado (cercano a 1) o en cambio es despreciable (cercano a 0).

Si además nos planteamos otros conceptos imprecisos como por ejemplo: Alejandro es alto, pero Ana no es bajita o la inflación actual aumenta rápidamente, etc… inmediatamente nos damos cuenta que hay un sinfín de conjuntos indefinidos que solo son aplicables a la computación si previamente son supervisados por la lógica difusa.

El objetivo que se persigue en este proyecto es imitar el razonamiento humano, por lo que el índice de fracaso en principio es muy elevado debido a que un computador necesita tener predefinidos todos y cada uno de los posibles inconvenientes que pueden surgir.

En Inteligencia Artificial, la Lógica Difusa o Lógica Fuzzy se utiliza para la resolución de una variedad de problemas, principalmente los relacionados con control de procesos industriales complejos y sistemas de decisión en general. Los sistemas basados en lógica difusa imitan la forma de tomar decisiones de los humanos, con la ventaja de ser mucho más rápidos.

La aplicación de la lógica difusa se realiza con la intención de imitar el razonamiento humano en la programación de computadoras. Con la lógica convencional, las computadoras pueden manipular valores estrictamente duales, como verdadero/falso, si/no o ligado/desligado. En la lógica difusa, se usan modelos matemáticos para mapear nociones subjetivas, como caliente/tibio/frio, para valores concretos que puedan ser manipulados por los ordenadores.

5.7.3. Proceso de razonamiento paso a paso

El resolvedor de problemas examina y recorre el árbol de manifestaciones, a partir del objeto raíz, hasta que se llega a la resolución del problema o a una situación en la que el resolvedor no es capaz de ofrecer un y solo un diagnóstico posible de entre todos los presentes en la base de hechos por no estar bien definida y/o no ser completa.

Puede suceder que la base de conocimientos no esté completa y que alguna manifestación no esté relacionada con algún diagnóstico o cualquier otra causa que determine que el resolvedor de problemas no sea capaz de obtener una solución o diagnóstico. De ahí la importancia de definir una base de conocimientos completa, de manera concienzuda y estableciendo las relaciones entre diagnósticos y manifestaciones con los valores apropiados en sus propiedades, tales como la frecuencia, la sugerencia, etc.

En el proceso de razonamiento podemos partir de dos situaciones:

- Una, en la que el usuario ha activado determinadas manifestaciones directamente en el árbol, activándose pues, éstas y sus descendientes (siempre y cuando alguna de sus descendientes no sea excluyente) y aumentando o disminuyendo el grado de activación de los diagnósticos relacionados con ellas. También podemos encontrarnos con que el usuario haya desactivado manifestaciones también directamente en el árbol, y que, por tanto, tanto estas, como sus descendientes, no van a tenerse en cuenta en el proceso de razonamiento.

- Otra, en la que el usuario no haya activado ninguna manifestación; situación, que, por otra parte, es la situación de partida con que el usuario se encuentra cuando inicia la aplicación. En este caso, el sistema, que se encuentra con todas las manifestaciones negadas (estado de partida de todas las manifestaciones y que permite que sean candidatas para ser preguntadas por su ocurrencia en el problema), deberá, en principio, recorrer todo el árbol de manifestaciones para ir descartando manifestaciones según el usuario vaya respondiendo a las preguntas que le vaya haciendo el resolvedor.

El resolvedor comenzará preguntando por las manifestaciones habituales, es decir, aquellas manifestaciones en cuya propiedad de relación con el diagnóstico tenga el valor *habitual = 'S'*. Se tendrá en cuenta el valor de la propiedad *coste* de la manifestación a la hora de elegir en qué orden se van realizando las preguntas de las manifestaciones, siendo las de menor coste las que se realizarán en primer lugar.

El mecanismo de razonamiento comenzará mostrando al usuario las preguntas de las manifestaciones habituales con menor coste y que se encuentren en estado negada (las manifestaciones con estado desactivada no se evaluarán y no se tendrán en cuenta en el problema a resolver).

Supongamos una situación de partida en la que se van a activar determinadas manifestaciones y se van a desactivar otras.

Supongamos que partimos de los siguientes supuestos:

- El paciente presenta dolor de cabeza.
- Tose a veces.

- No ha tenido vómitos.

La manifestación *tose a veces* es un claro ejemplo de un síntoma confuso, a priori, para el sistema. Es algo subjetivo, difícil de cuantificar. No está perfectamente definido si el paciente tiene tos o no tiene tos. Es un valor que solo puede ser tratado por el sistema gracias a la lógica difusa y al modelo matemático implementado.

Ante estas manifestaciones, el usuario debería activar la manifestación *dolor de cabeza* y la manifestación *tose a veces*. Por otro lado, debería desactivar la manifestación *vómitos*.

En la figura 5.2 podemos observar la representación de estas tres manifestaciones en el árbol.

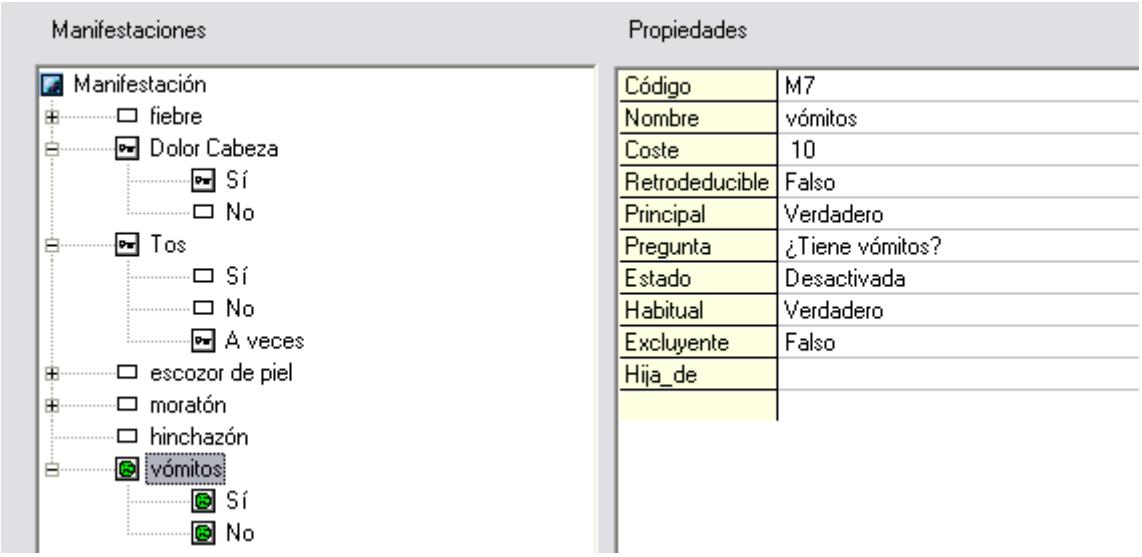


Figura 5.2. Representación de las manifestaciones

En la figura 5.2 observamos que, tras desactivar la manifestación *vómitos*, sus dos manifestaciones descendientes *Sí* y *No* se han desactivado también y, por tanto, no serán tenidas en cuenta cuando se inicie el proceso de resolución del problema.

También podemos comprobar cómo las manifestaciones *Dolor Cabeza Sí* y *Tos A veces* se han activado.

¿Qué consecuencias se derivan de la activación de estas manifestaciones, por un lado, y la desactivación de la manifestación *vómitos*, por otro, en cuánto al escenario inicial del que va a partir el resolvedor de problemas?. Veámoslo con detalle.

La manifestación *Dolor Cabeza* está relacionada con el diagnóstico *gripe* y el valor de la propiedad sugerencia es 6. Además es una relación positiva, por lo que el grado de activación del diagnóstico *gripe* aumentará en 6, tras la activación de la manifestación.

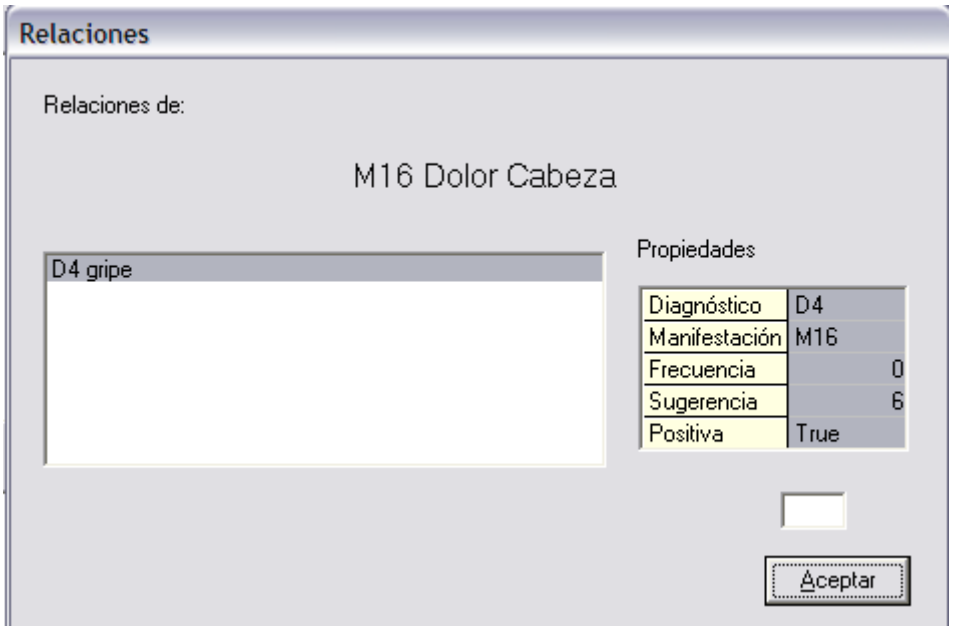


Figura 5.3. Relación *Dolor Cabeza <-> gripe*

La manifestación descendiente *Sí* de la manifestación padre *Dolor Cabeza* también está relacionada con el diagnóstico *gripe* con valor 2 en la propiedad sugerencia y también es positiva, por lo que el grado de activación del diagnóstico *gripe* aumentará en 2.

Por lo tanto, tras la activación de la manifestación *Dolor Cabeza* y de su descendiente *Sí,* el grado de activación del diagnóstico *gripe* ha aumentado en 8. Si tenemos en cuenta que previamente tenía un grado de activación igual a 0, el diagnóstico *gripe* quedaría con un grado de activación igual a 8.

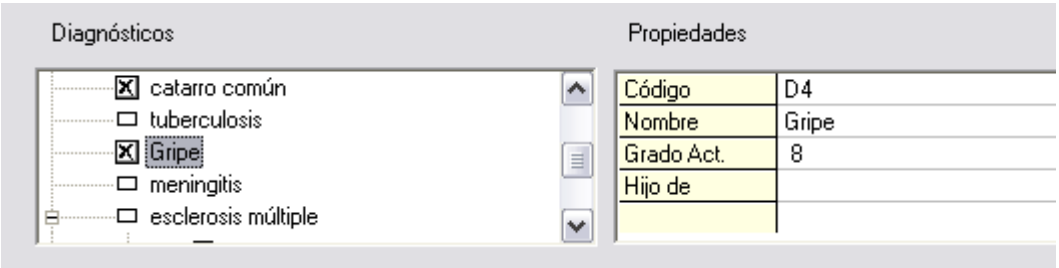


Figura 5.4. Grado de activación del diagnóstico *gripe*

Por otro lado, la manifestación *Tos* está relacionada con los diagnósticos *Resfriado Común* y *gripe*.

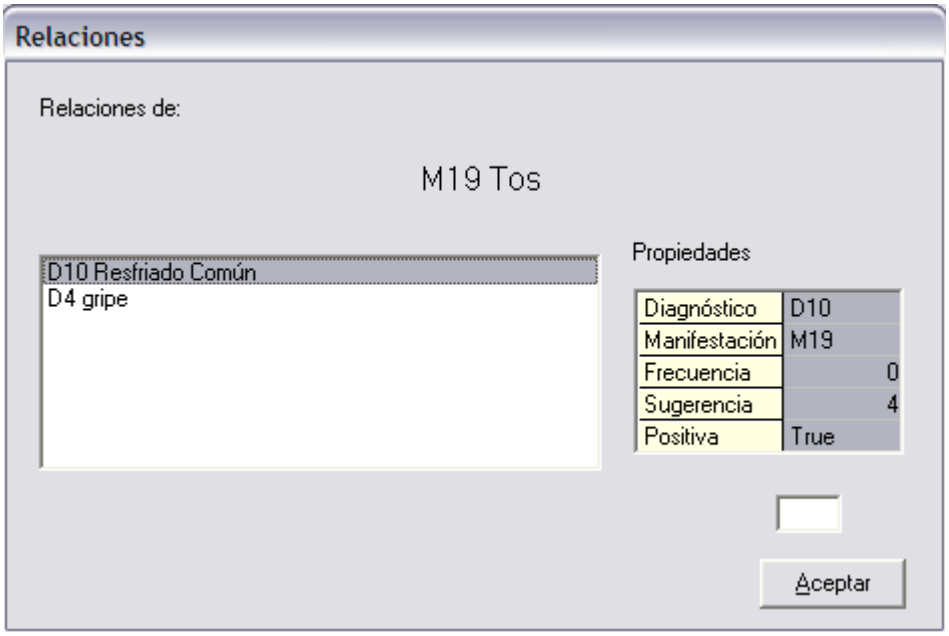


Figura 5.5. Relación *Tos* <-> *Resfriado Común*

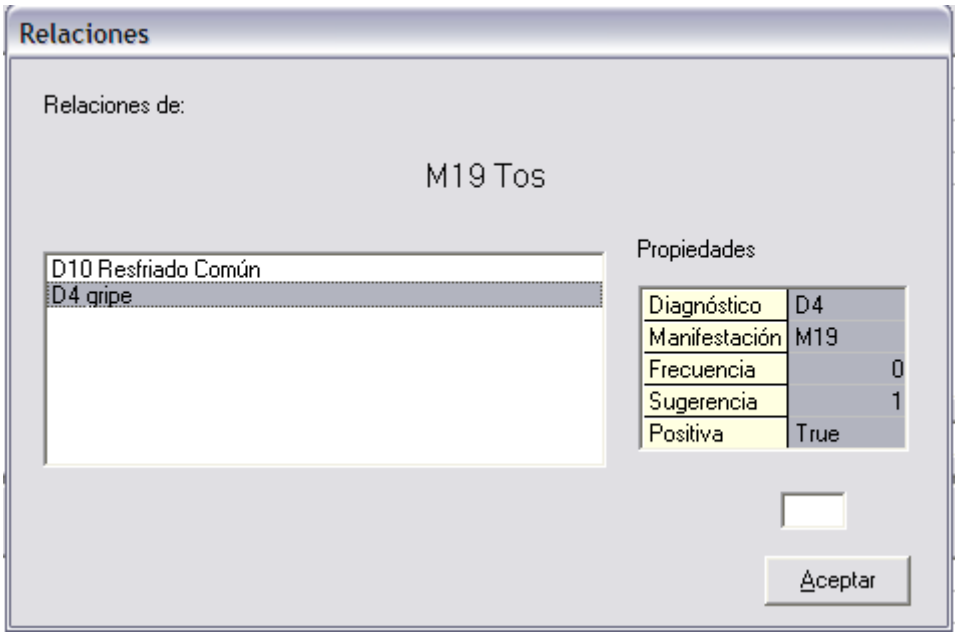


Figura 5.6. Relación *Tos* <-> *gripe*

En las figuras 5.5 y 5.6 se muestran los valores de la propiedad sugerencia de las relaciones de la manifestación *Tos* con los diagnósticos *Resfriado Común* y *gripe*. La activación de la manifestación *Tos* aumentará en 4 el grado de activación del diagnóstico *Resfriado Común* y en uno el del diagnóstico *gripe*.

La manifestación *A veces*, descendiente de la manifestación *Tos*, también se ha activado y está relacionada también con los diagnósticos *Resfriado Común* y *gripe*.

En la figura 5.7 se muestra la relación existente entre la manifestación *A veces* y el diagnóstico *Resfriado Común*. Vemos que es una relación positiva y sugiere al diagnóstico con un valor 2, lo que hará que aumente en esa cantidad su grado de activación.

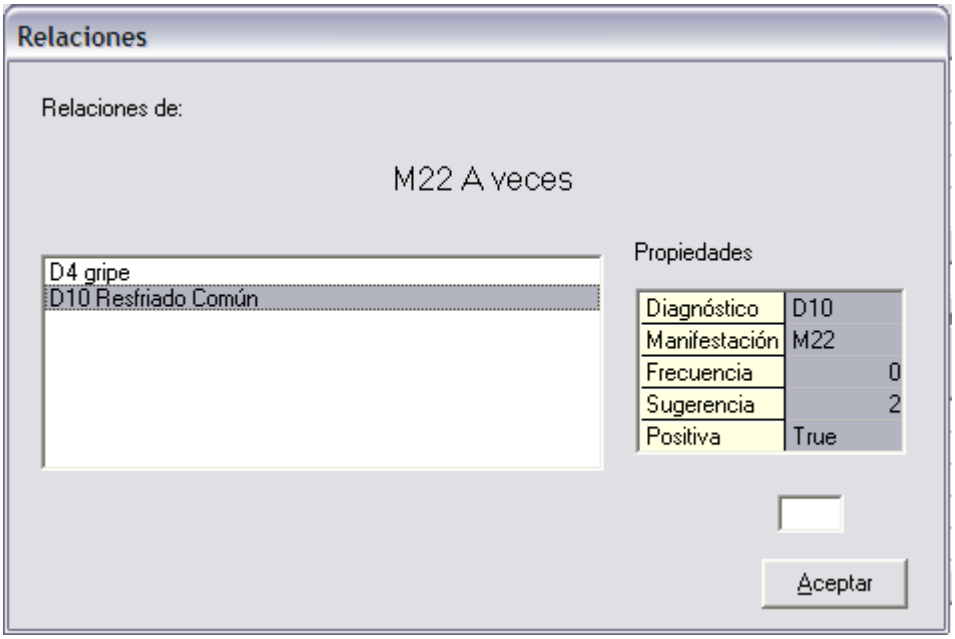


Figura 5.7. Relación *A veces* <-> *Resfriado Común*

Sin embargo, como se puede apreciar en la figura 5.8, la relación entre la manifestación *A veces* y el diagnóstico *gripe* es negativa y el valor de la propiedad frecuencia es de 3 con lo que descenderá en esa cantidad el grado de activación del diagnóstico *gripe*.

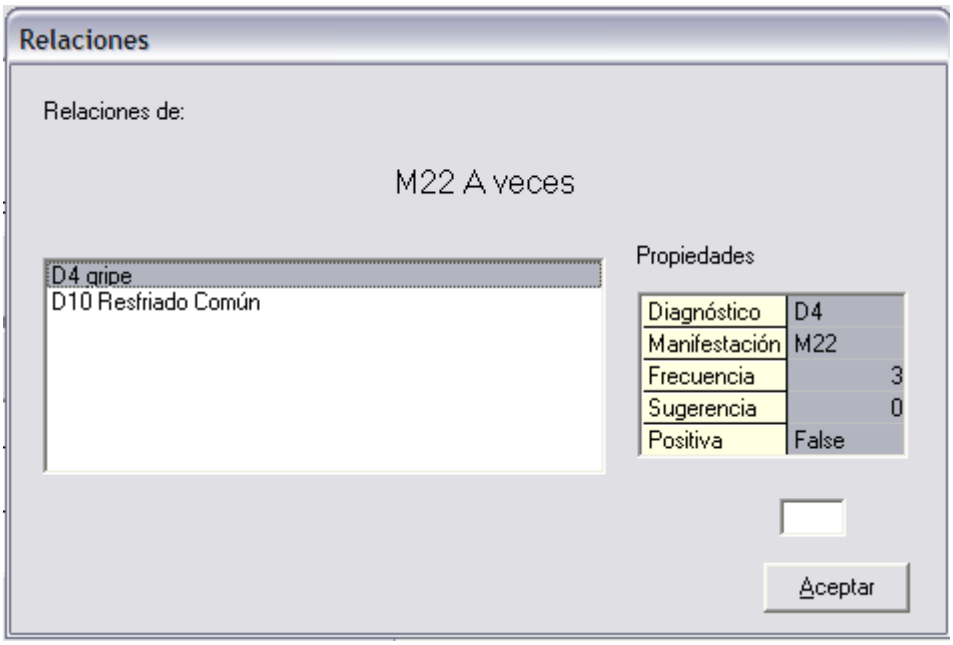


Figura 5.8. Relación *A veces <-> gripe*

Finalmente, los grados de activación de los diagnósticos implicados en la activación, por parte del usuario, de las manifestaciones descritas, y que serán punto de partida para el resolvedor de problemas, quedarían con los siguientes valores:

Grado de activación del diagnóstico *gripe*: 6 + 2 + 1 – 3 = 6

Grado de activación del diagnóstico *Resfriado Común*: 4 + 2 = 6

Casualmente, ambos diagnósticos han quedado activados en el mismo grado tras la activación de las manifestaciones *Dolor de cabeza* y *Tos* por parte del usuario.

Por lo tanto, esta será la situación de partida para el resolvedor de problemas, una vez el usuario inicie el proceso razonamiento mediante la opción correspondiente del menú de la aplicación.

El resolvedor de problemas comienza el mecanismo de razonamiento desde el objeto raíz (figura 5.9) como raíz del árbol causal de manifestaciones.

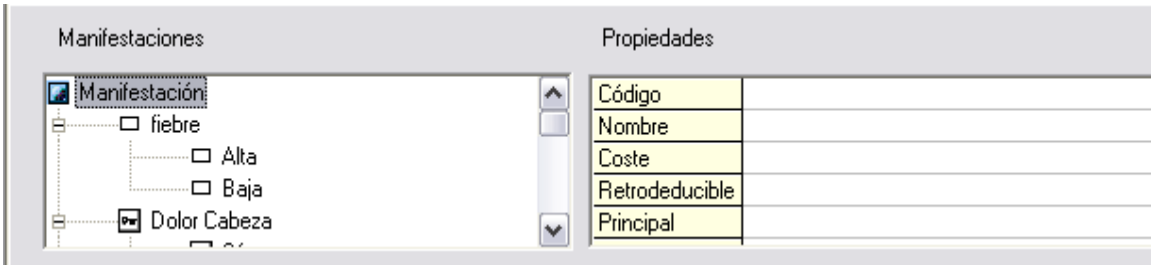


Figura 5.9. Manifestación raíz del árbol de manifestaciones

La manifestación raíz no se evalúa y no tiene valores en sus propiedades. Tan sólo es el punto de partida del resolvedor de problemas y sirve como objeto raíz del árbol.

El mecanismo de razonamiento continuaría evaluando la manifestación *fiebre* debido a que es una manifestación habitual, tiene el valor negada en la propiedad Estado y tiene un coste 0, por lo que es candidata a ser la primera manifestación por la que preguntar al usuario.

Por lo tanto, al cumplir estas características y, sobre todo, estar negada, el mecanismo seguirá por esta manifestación y sus descendientes, ya que, si estuviera desactivada no se evaluaría y tampoco las manifestaciones descendientes. Tampoco la evaluaría si se encontrase activada, pues indica que ya ha sido tratada, bien por el propio resolvedor, bien por parte del usuario activandola manualmente.

Como se puede observar en la figura 5.10 la manifestación *fiebre* tiene valor falso en la propiedad retrodeducible, ya que de dicha

manifestación no podemos deducir ninguna otra. Tiene valor verdadero en la propiedad principal, por lo que se realizará al usuario la pregunta que aparezca en la propiedad pregunta de la manifestación. En este caso *¿Tiene fiebre?*.

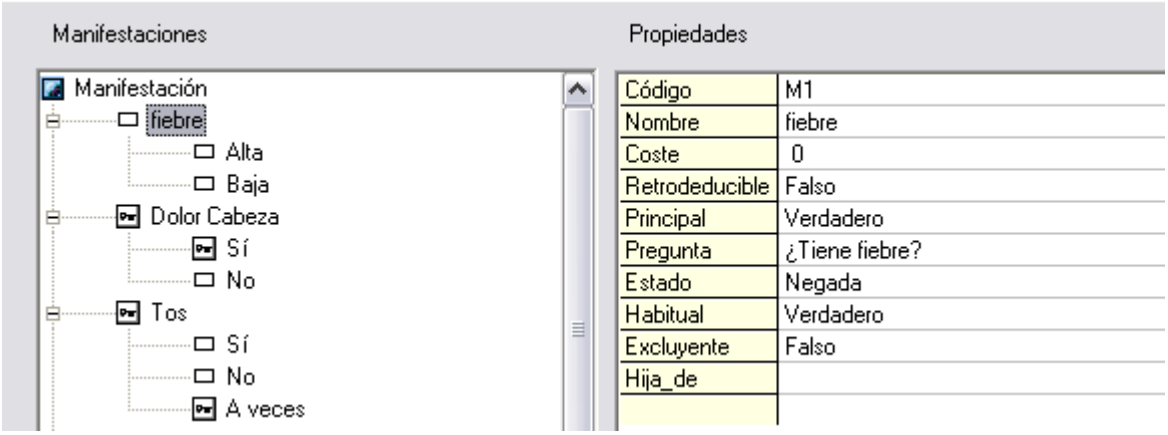


Figura 5.10. Manifestación *fiebre* del árbol de manifestaciones

Figura 5.11. Pregunta de la manifestación *fiebre* que se le hace al usuario

Si el usuario contesta afirmativamente a la pregunta *¿Tiene fiebre?*, la manifestación pasaría a estado activada, y esto afectaría al grado de activación de los diagnósticos con los que está relacionada la manifestación *fiebre*, aumentando el grado de activación de estos, si la relación es positiva, en cuyo caso el aumento sería equivalente al valor de la propiedad sugerencia de la relación, o bien, disminuyendo el grado de activación de los diagnósticos relacionados con la manifestación, si la relación es negativa y, en cuyo caso, la disminución sería equivalente al valor de la propiedad frecuencia de la relación.

Hay que recordar que las manifestaciones activas positivas aumentan el grado de activación de los diagnósticos mediante la propiedad sugerencia de la relación y que las manifestaciones activas negativas disminuyen el grado de activación de los diagnósticos mediante la propiedad frecuencia de la relación.

Los diagnósticos relacionados con la manifestación *fiebre* son *Resfriado Común* y *Tipo A*. En las figuras 5.12 y 5.13 se puede apreciar las

propiedades de ambas relaciones: *fiebre <-> Resfriado Común* y *fiebre <-> Tipo A*. Podemos comprobar cómo la manifestación *fiebre* sugiere el diagnóstico *Resfriado Común* con un valor de 2 (en la propiedad de relación sugerencia) y, a su vez, se estima una frecuencia de 3 (en la propiedad de relación frecuencia) con el diagnóstico *Tipo A*.

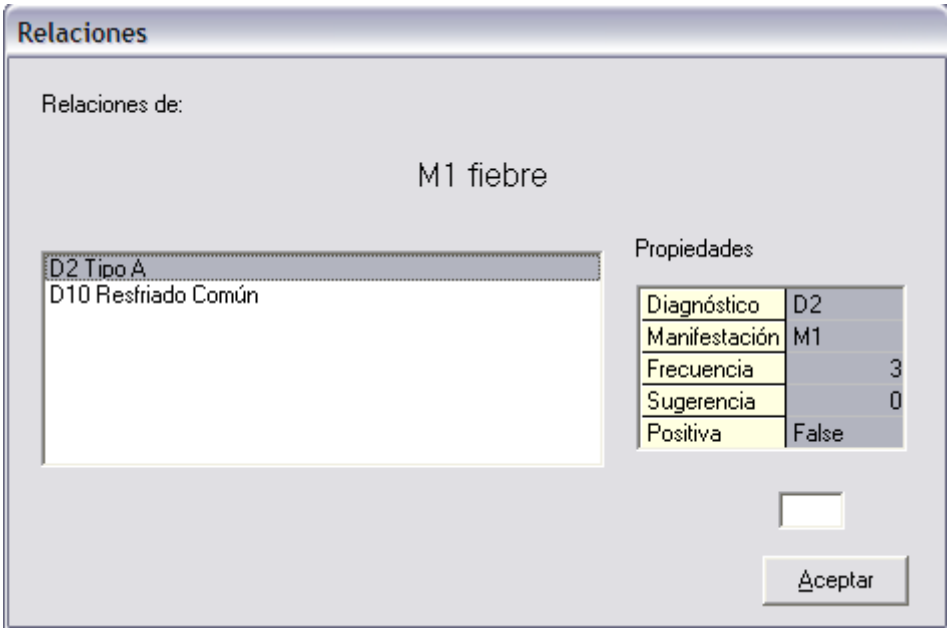


Figura 5.12. Propiedades de la relación *fiebre <-> Resfriado Común*

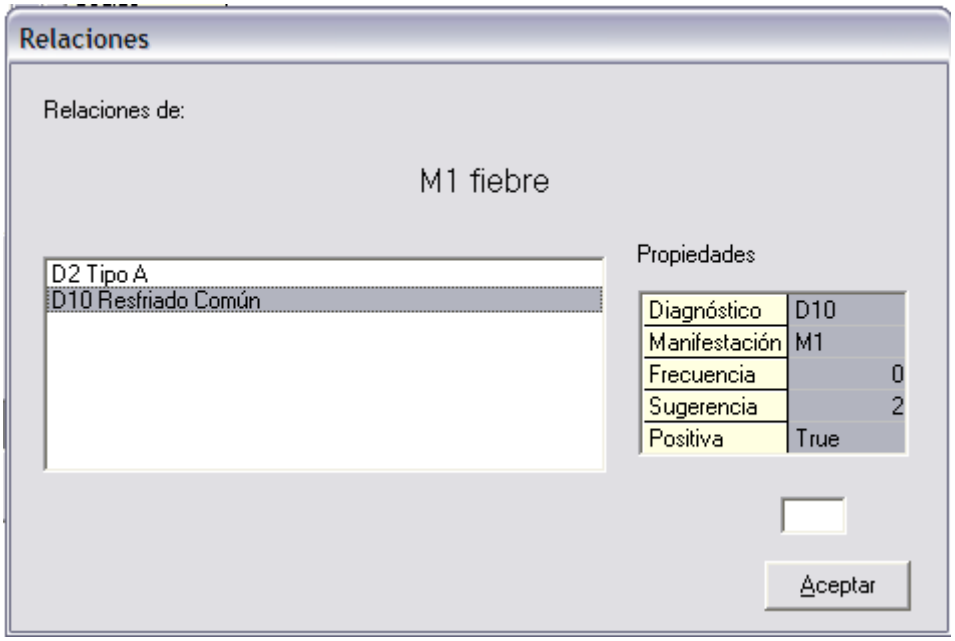


Figura 5.13. Propiedades de la relación *fiebre <-> Tipo A*

Los grados de activación de los diagnósticos *Resfriado Común* y *Tipo A* relacionados con la manifestación fiebre antes de activarla se pueden apreciar en las figuras 5.14 y 5.15 respectivamente.

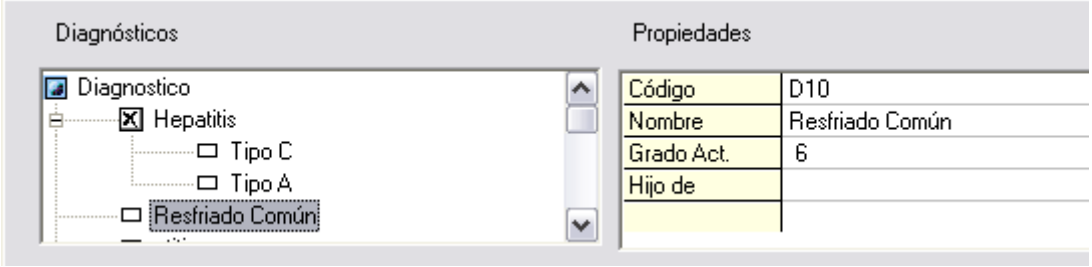


Figura 5.14. Grado de activación del diagnóstico *Resfriado Común*

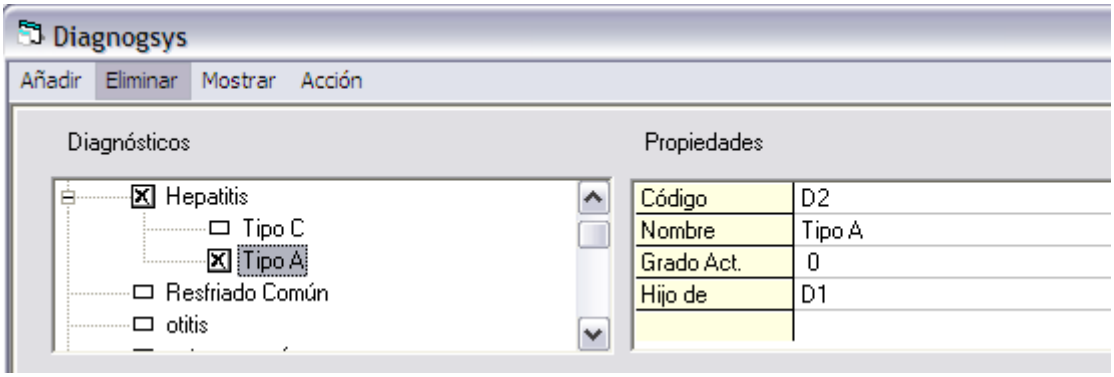


Figura 5.15. Grado de activación del diagnóstico *Tipo A*

Tras la activación de la manifestación *fiebre* y al contestar el usuario afirmativamente a la pregunta *¿tiene fiebre?*, los diagnósticos relacionados han aumentado o disminuido su grado de activación teniendo en cuenta el valor de las propiedades sugerencia y frecuencia de los diagnósticos relacionados con la manifestación *fiebre*.

En este caso en concreto, el grado de activación del diagnóstico *Resfriado Común* pasaría de tener valor 6 a tener valor 8, y el grado de activación del diagnóstico *Tipo A* pasaría de tener un valor de 0 a un valor igual a -3.

El resolvedor de problemas continuaría con la siguiente manifestación recorriendo el árbol en amplitud, con lo que examinaría la manifestación *Alta*, ya que la activación de la manifestación *fiebre* provoca la activación de sus descendientes. En este caso, la activación no es directa debido a que la manifestación *Alta* es excluyente y, por tanto, puede activarse ella o la otra manifestación hija de la manifestación *fiebre*, la manifestación *Baja*.

Esta manifestación está negada, con lo que el resolvedor la tendrá en cuenta. Es secundaria, y por tanto, no tiene valor en la propiedad pregunta, de ahí que se le haga al usuario la pregunta completa de la manifestación padre y la manifestación en cuestión. Esto es, se le preguntará *¿Tiene fiebre/Alta?*, como se puede observar en la figura 5.16.

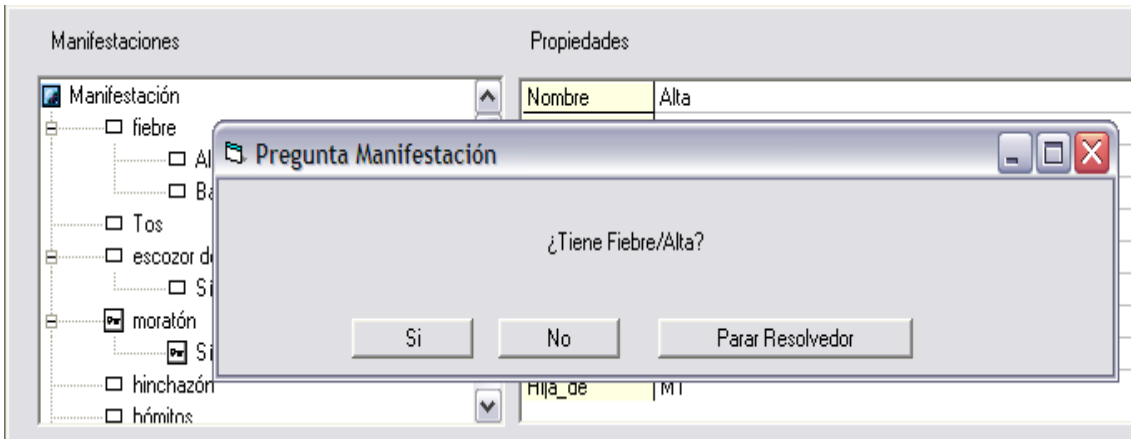


Figura 5.16. Pregunta que se le hace al usuario, resultado de la pregunta de la manifestación padre y de la manifestación *Alta*

Si el usuario contesta afirmativamente a la pregunta *¿Tiene fiebre/Alta?* la manifestación *Alta* pasaría a estado activada y esto afectaría al grado de activación de los diagnósticos con los que está relacionada la manifestación.

En este caso, la manifestación está relacionada con el diagnóstico *gripe* y le sugiere con un valor igual a 3, lo que aumentará en dicho valor el grado de activación del diagnóstico *gripe*.

Hasta este momento de la descripción en detalle del proceso de razonamiento, tenemos tres diagnósticos implicados y posibles candidatos a ser la solución al problema. Estos son: *Resfriado Común*, *Tipo A* y *gripe*.

El diagnóstico *Resfriado Común* ha aumentado su grado de activación hasta el valor 8, el diagnóstico *Tipo A* ha disminuido su grado de activación al valor -3 y, por último, el diagnóstico *gripe* ha pasado de tener un grado de activación con valor 6 a tener un valor igual a 9.

El mejor candidato a ser solución del problema es, por tanto, el diagnóstico *gripe*, ya que, hasta el momento, es el diagnóstico con mayor grado de activación de la base de hechos.

Como se puede ver en la Figura 5.16 cuando se realiza la pregunta de la manifestación que el resolvedor está evaluando, nos encontramos con un botón con el texto *Parar Resolvedor*.

En cualquier momento del proceso de razonamiento el usuario puede detenerlo si pulsa sobre el botón *Parar Resolvedor*. Una vez hecho esto, y detenido el proceso de razonamiento, los grados de activación de los diagnósticos tendrían los valores que tuvieran hasta ese momento.

Si el usuario seleccionase la opción de menú **Acción -> Resolver** el sistema intentaría dar una única solución al problema y, con los valores de los grados de activación de los diagnósticos que tuvieran en el momento de detener el usuario la ejecución del proceso de razonamiento, tratar de dar

un diagnóstico o, por el contrario, mostrar al usuario un mensaje indicando que no es posible resolver el problema con los datos que se disponen.

En nuestro ejemplo, si el usuario decidiese detener el resolvedor, nos encontraríamos con los siguientes grados de activación de los diagnósticos implicados:

Grado de activación del diagnóstico *gripe*: 6 + 3 = 9

Grado de activación del diagnóstico *Tipo A: -3*

Grado de activación del diagnóstico *Resfriado Común*: 6 + 2 = 8

El sistema mostraría al usuario la solución al problema indicando que el diagnóstico es gripe, como se puede apreciar en la Figura 5.17.

Figura 5.17. El diagnóstico *gripe* es la solución al problema

Si el usuario no hubiese decidido detener el proceso de razonamiento y continuar, el resolvedor continuaría evaluando las siguientes manifestaciones, con los consiguientes aumentos y disminuciones de los grados de activación de los diagnósticos con los que se relacionan, hasta evaluar la última manifestación negada de la base de hechos, momento en el que seleccionaría el diagnóstico con mayor grado de activación.

En el supuesto en el que el usuario hubiese decidido parar el resolvedor y el sistema no fuese capaz de obtener una solución al problema, y existiesen aún manifestaciones negadas por las que el resolvedor pudiese preguntar, se podría continuar con el proceso de razonamiento en la situación en que se encontrara la base de hechos antes de parar el proceso de resolución del problema. Bastaría con seleccionar la opción de menú Acción -> Iniciar Resolvedor.

Por lo tanto, el usuario puede detener y volver a iniciar el proceso de resolución del problema, tantas veces determine oportuno, según se encuentre con unas situaciones u otras, y así poder obtener un diagnóstico temprano cuando considere que las manifestaciones activadas por el resolvedor son lo bastante aclaratorias como para considerar que ya está más o menos claro el diagnóstico.

Es una posibilidad que ofrece el sistema y queda en manos del usuario utilizar esta estrategia o no para encontrar una solución al problema, aunque no es la más aconsejable para usuarios inexpertos o no conocedores del ámbito médico en cuestión.

La estrategia habitual a seguir, y más aconsejable, es permitir al resolvedor de problemas terminar la ejecución del proceso de razonamiento y que sea este quien nos proporcione un diagnóstico final, una vez agotadas todas las posibilidades de activación o desactivación de las manifestaciones presentes en la base de hechos.

Detener el resolvedor de problemas, para luego continuar con el proceso de resolución, puede utilizarse también para activar o desactivar manualmente alguna manifestación que, o bien el paciente no nos había manifestado inicialmente y lo hace ahora, o el usuario considera que debe activarse por estar presente claramente en el problema a resolver una vez que ya han quedado activadas otras manifestaciones.

Esta posibilidad parece más útil de lo que pueda parecer inicialmente, pues el sistema permite avanzar con el proceso de resolución del problema “a tramos”, pudiendo manipular las manifestaciones en cualquier momento del proceso, incluso modificar alguno de los valores de las propiedades de la manifestaciones, diagnósticos o relaciones, y continuar con el mecanismo de resolución.

6. MANUAL DE USUARIO

6.1. Introducción

En las siguientes secciones y subsecciones se describirá el entorno de la aplicación, así como todas aquellas operaciones que el usuario podrá realizar gracias a las opciones del menú principal y a las opciones de menús emergentes y contextuales que permiten facilitar al usuario la realización de las operaciones más habituales.

6.2. Entorno de la aplicación

Al iniciarse la aplicación podemos observar que en la ventana principal se distinguen tres zonas: la ventana principal, con ciertas opciones de menú, que engloba a dos secciones, muy similares entre sí y que están delimitadas por la ventana principal.

En la ventana principal podremos realizar, gracias a las opciones del menú, operaciones que van a afectar a elementos de las otras dos ventanas.

Una de las dos secciones mostrará información sobre los diagnósticos presentes en la base de datos y la otra mostrará información sobre las manifestaciones. Esta información es la que el usuario ha almacenado en la base de datos como se describirá en secciones posteriores.

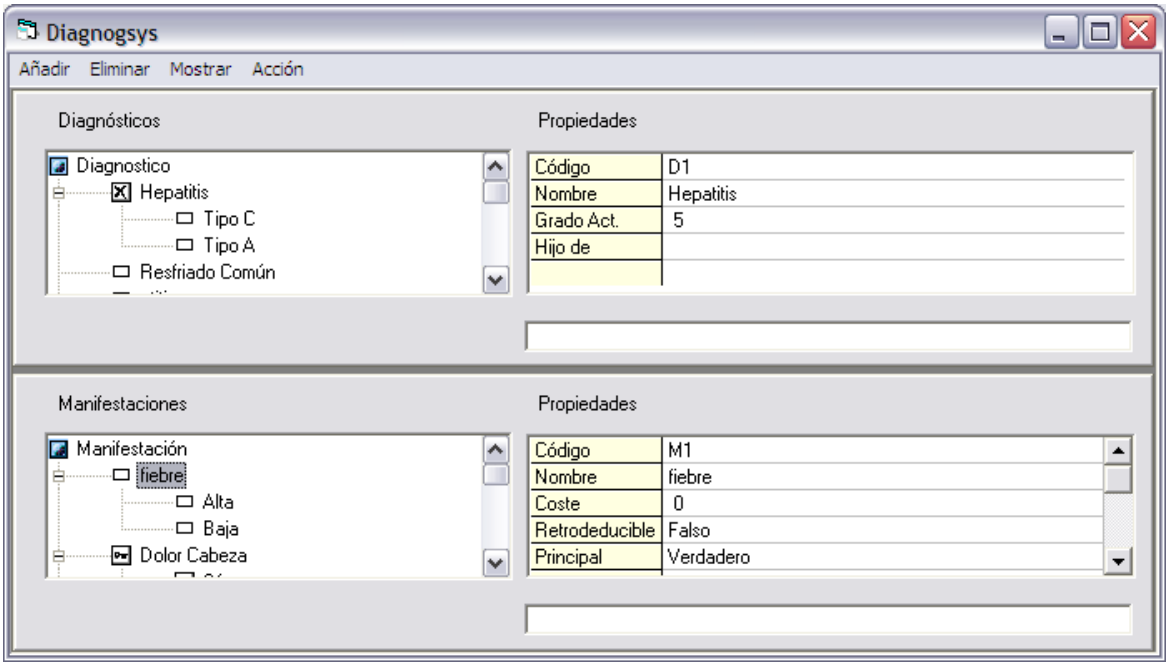


Figura 6.1. Ventana principal y secciones de diagnósticos y manifestaciones.

6.3. Árbol de Diagnósticos

Es en esta sección de la interfaz gráfica donde el usuario podrá realizar las operaciones relacionadas con los diagnósticos. Operaciones tales como inserción de nuevos diagnósticos, eliminación, modificación de sus propiedades, ver las relaciones que tiene con las manifestaciones, etc.

En primer término, en la parte izquierda, podemos apreciar una estructura jerárquica en forma de árbol en la que se presentarán los distintos diagnósticos que están presentes en la base de datos. Esta estructura representa los diagnósticos de forma jerárquica, es decir, se muestra la dependencia existente entre ellos.

Un diagnóstico depende de otro cuando éste está representado a un nivel inferior. En la Figura 6.2 se puede apreciar la estructura jerárquica en forma de árbol de los diagnósticos.

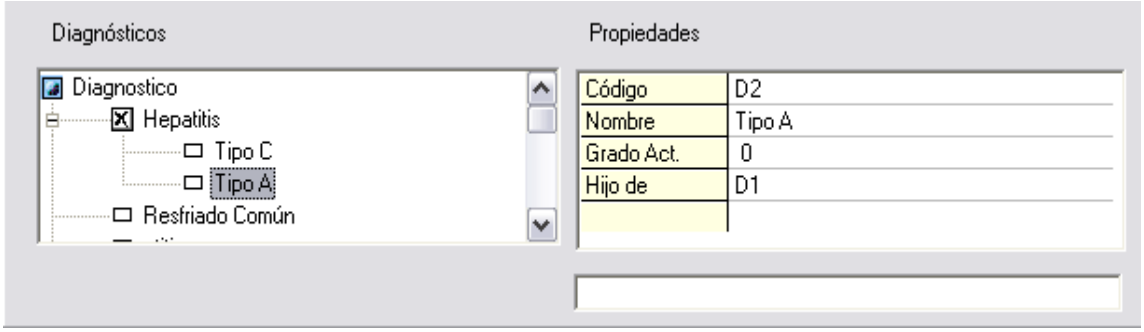


Figura 6.2. Ventana con el árbol de diagnósticos y las propiedades de éstos.

En la Figura 6.2 podemos observar la dependencia jerárquica de los diagnósticos. Se puede apreciar cómo el diagnóstico Hepatitis es padre del diagnóstico Tipo A.

También podemos apreciar en esta figura, en la parte derecha, la manera en la que se muestran las propiedades del diagnóstico seleccionado en el árbol (el que se muestra resaltado).

Además se aprecia, en el ángulo inferior derecho, un espacio reservado para que el usuario pueda modificar las propiedades del diagnóstico seleccionado en el árbol. Si pinchamos con el ratón sobre cualquiera de las propiedades del diagnóstico (excepto la del código y la de Hijo de que no son modificables) se permitirá al usuario introducir un nuevo valor para la propiedad elegida.

6.3.1. Añadir diagnóstico

Para añadir un diagnóstico es necesario seleccionar el diagnóstico del que queramos que dependa (si no se desea que dependa de ningún otro, deberemos seleccionar la raíz del árbol, diferenciada del resto de los nodos con un icono diferente con el texto "Diagnostico") y seleccionar la opción del menú principal Añadir → Diagnóstico, tal y como se puede observar en la Figura 6.3.

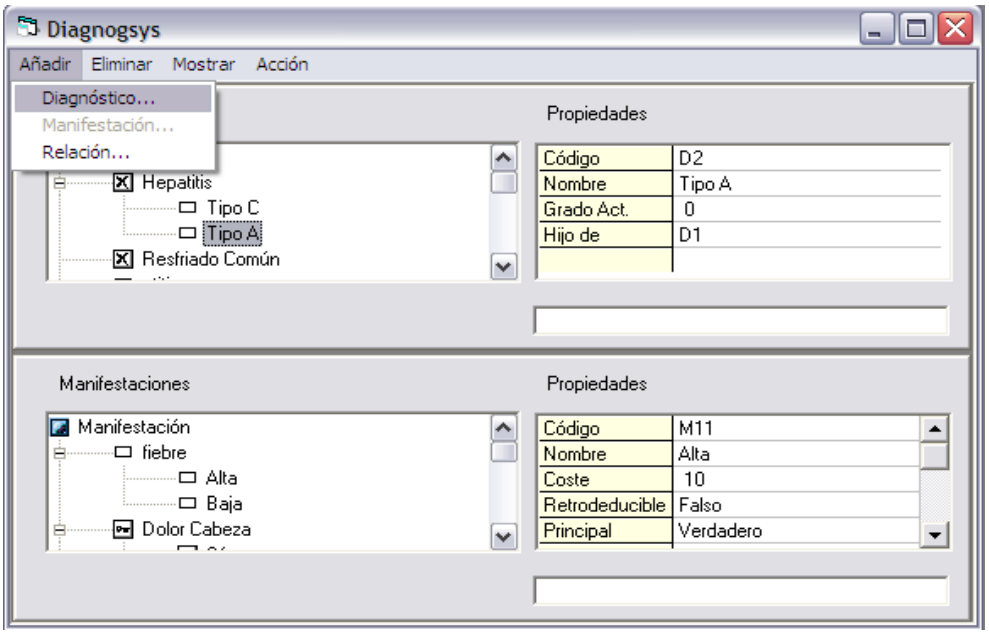


Figura 6.3. Opción Añadir Diagnóstico del menú principal.

También se puede añadir un diagnóstico seleccionando el diagnóstico del que queramos que dependa y pulsando el botón derecho del ratón seleccionando la opción Añadir Diagnóstico, como se muestra en la Figura 6.4.

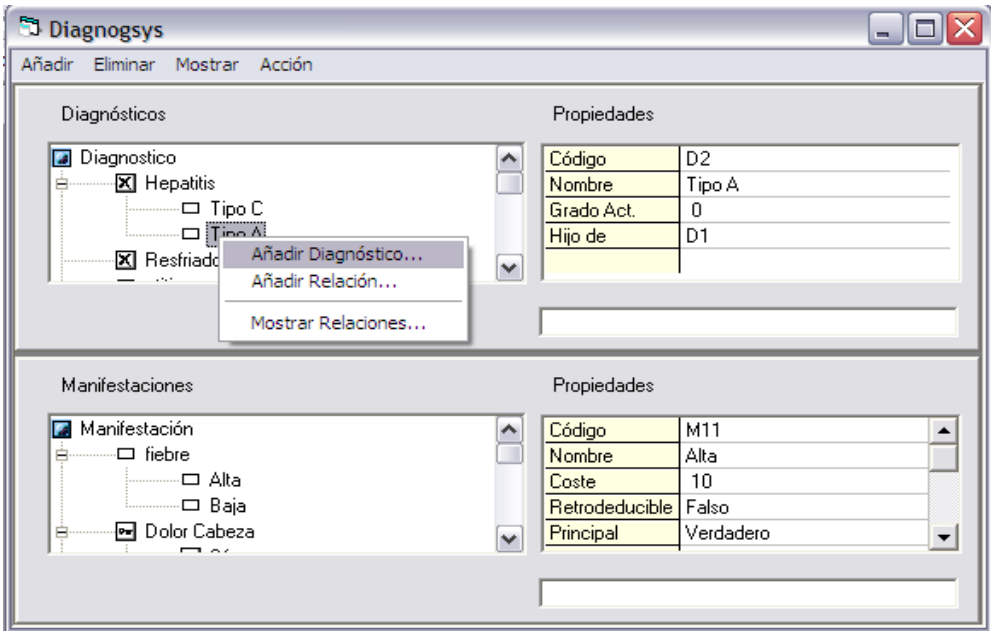


Figura 6.4. Opción Añadir Diagnóstico del menú emergente.

Una vez seleccionada la opción Añadir Diagnóstico, por un método u otro, aparecerá una ventana que pedirá al usuario el nombre del nuevo diagnóstico.

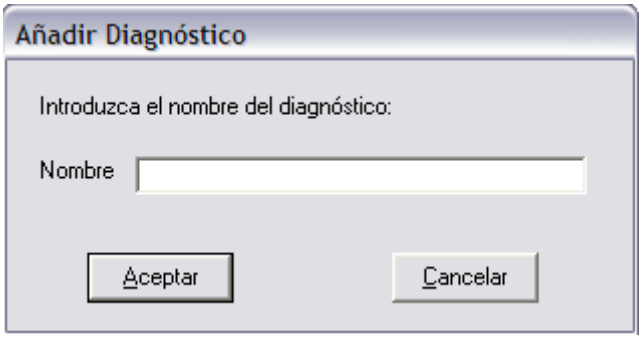


Figura 6.5. Ventana de solicitud del nombre del nuevo diagnóstico.

Una vez que el usuario pulse el botón Aceptar, el nuevo diagnóstico se almacenará en la base de datos y se mostrará en el árbol. En las propiedades, concretamente en la propiedad Hijo de, podremos observar cómo aparece el nombre del diagnóstico padre, si es que éste no es la raíz, en cuyo caso esta propiedad aparecerá vacía.

6.3.2. Eliminar diagnóstico

Para eliminar un diagnóstico deberemos seleccionarlo y pulsar la opción del menú principal Eliminar → Diagnóstico.

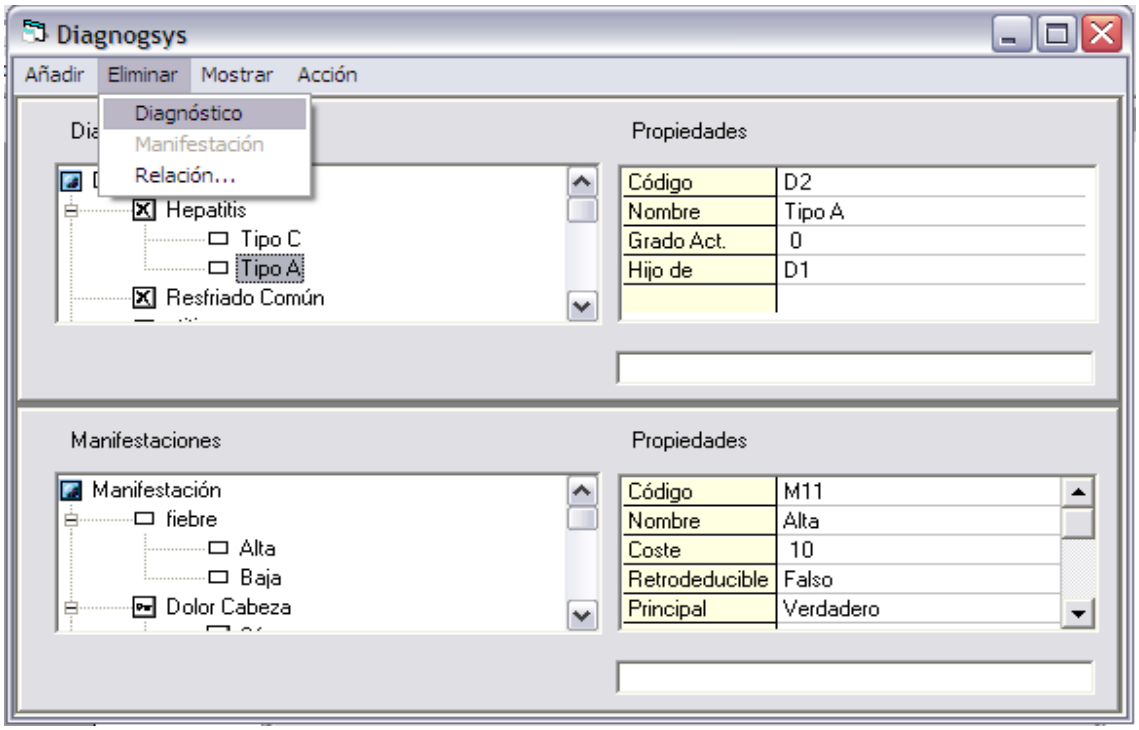


Figura 6.6. Opción de menú para eliminar el diagnóstico seleccionado.

El usuario debe tener en cuenta que al eliminar un diagnóstico se eliminarán todos aquellos que dependan de éste, es decir, se eliminarán todos los diagnósticos hijos del diagnóstico que vamos a eliminar, así como las relaciones que pudieran tener con las manifestaciones. Por este motivo aparecerá una ventana que advertirá al usuario de esta situación y le pedirá confirmación.

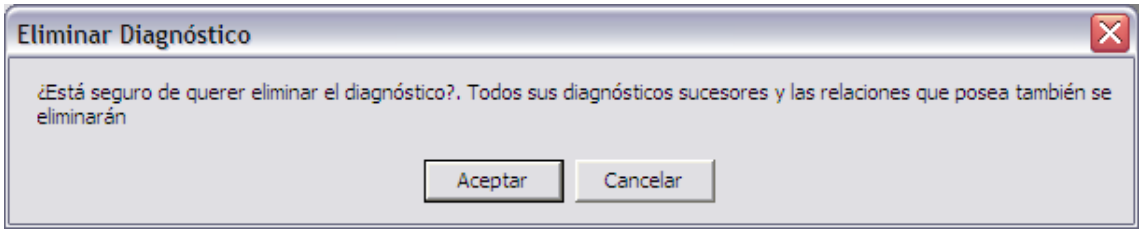


Figura 6.7. Ventana de advertencia de eliminación de diagnóstico.

Una vez eliminado el diagnóstico, él y todos sus hijos, si los tuviera, desaparecerán del árbol y de la base de datos, así como las relaciones que pudieran tener.

6.3.3. Modificar diagnóstico

Para modificar alguna de las propiedades de un diagnóstico deberemos seleccionarlo en el árbol y pinchar la propiedad que deseemos modificar en el cuadro de propiedades. A continuación se deberá introducir el nuevo valor en el recuadro habilitado para esta acción y pulsar la tecla ENTER.

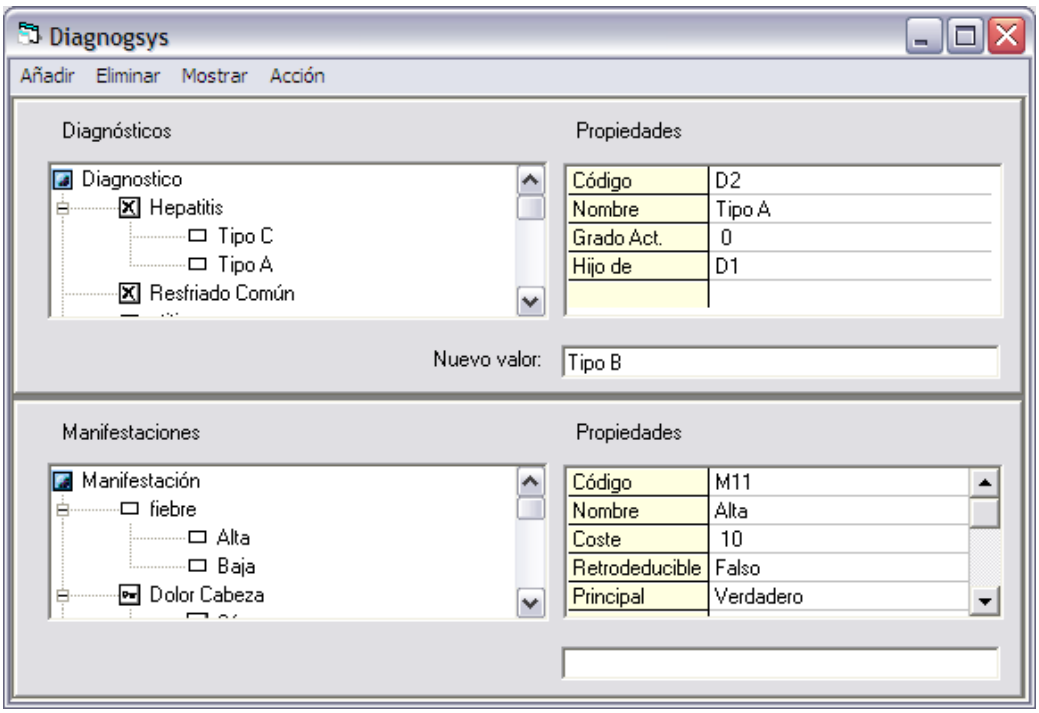


Figura 6.8. Modificando el nombre del diagnóstico Tipo A.

6.4. Árbol de Manifestaciones

La dinámica del árbol de manifestaciones y su funcionamiento es muy similar al de diagnósticos. Las operaciones de inserción, eliminación y modificación de las manifestaciones son análogas a las operaciones que se pueden realizar con los diagnósticos.

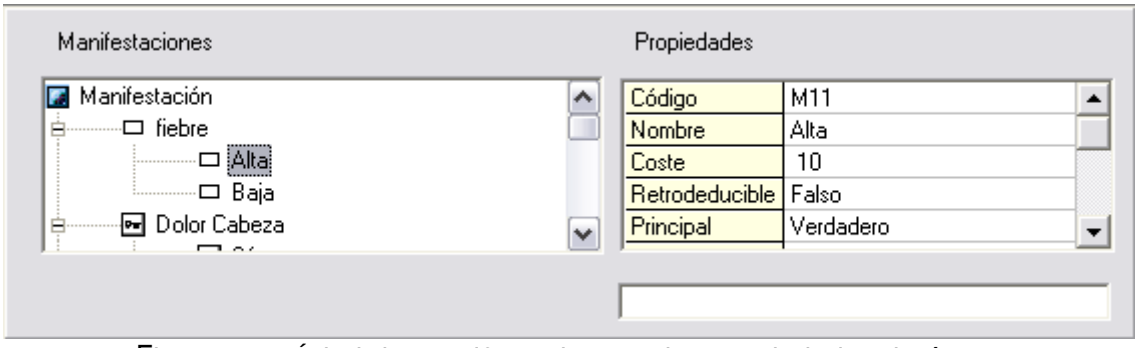


Figura 6.9. Árbol de manifestaciones y las propiedades de éstas.

6.4.1. Añadir manifestación

Para añadir una manifestación el usuario deberá seleccionar la manifestación de la que desee que dependa (si no se desea que dependa de ninguna otra, deberá seleccionar la raíz del árbol, diferenciada del resto de los nodos con un icono diferente con el texto "Manifestación") y seleccionar la opción del menú principal Añadir → Manifestación, tal y como se puede observar en la Figura 6.10.

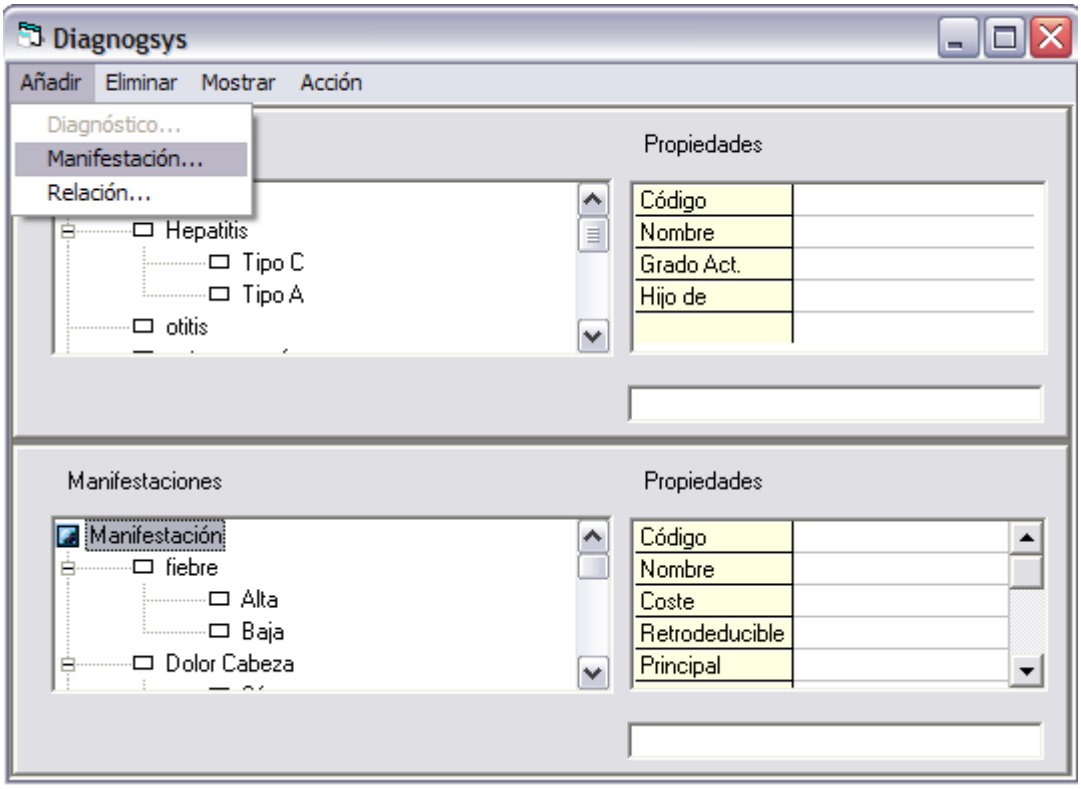


Figura 6.10. Opción Añadir Manifestación del menú principal.

También se puede añadir una manifestación seleccionando la manifestación de la que se desee que dependa y pulsando el botón derecho del ratón seleccionando la opción Añadir Manifestación, como se muestra en la Figura 6.11.

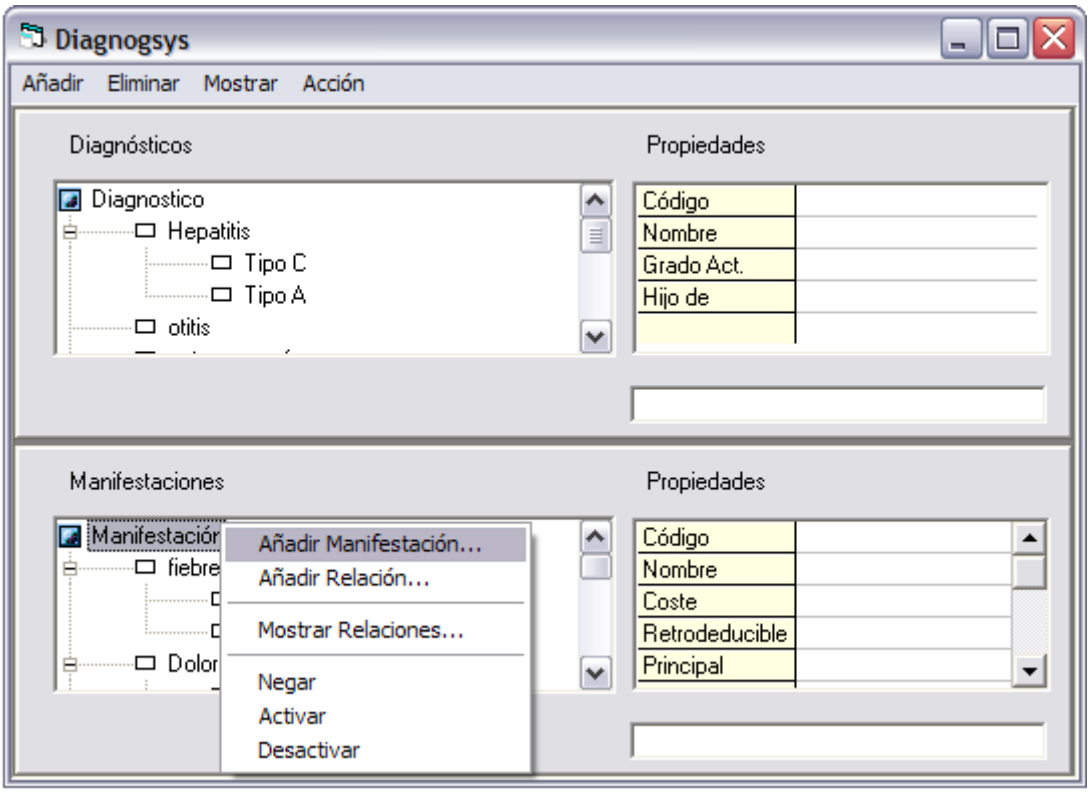


Figura 6.11. Opción Añadir Manifestación del menú emergente.

Una vez se seleccione la opción Añadir Manifestación aparecerá una ventana que pedirá al usuario el nombre de la nueva manifestación, el coste, el texto de la propiedad pregunta y si es una manifestación excluyente, retrodeducible y/o habitual, tal y como se puede observar en la Figura 6.12.

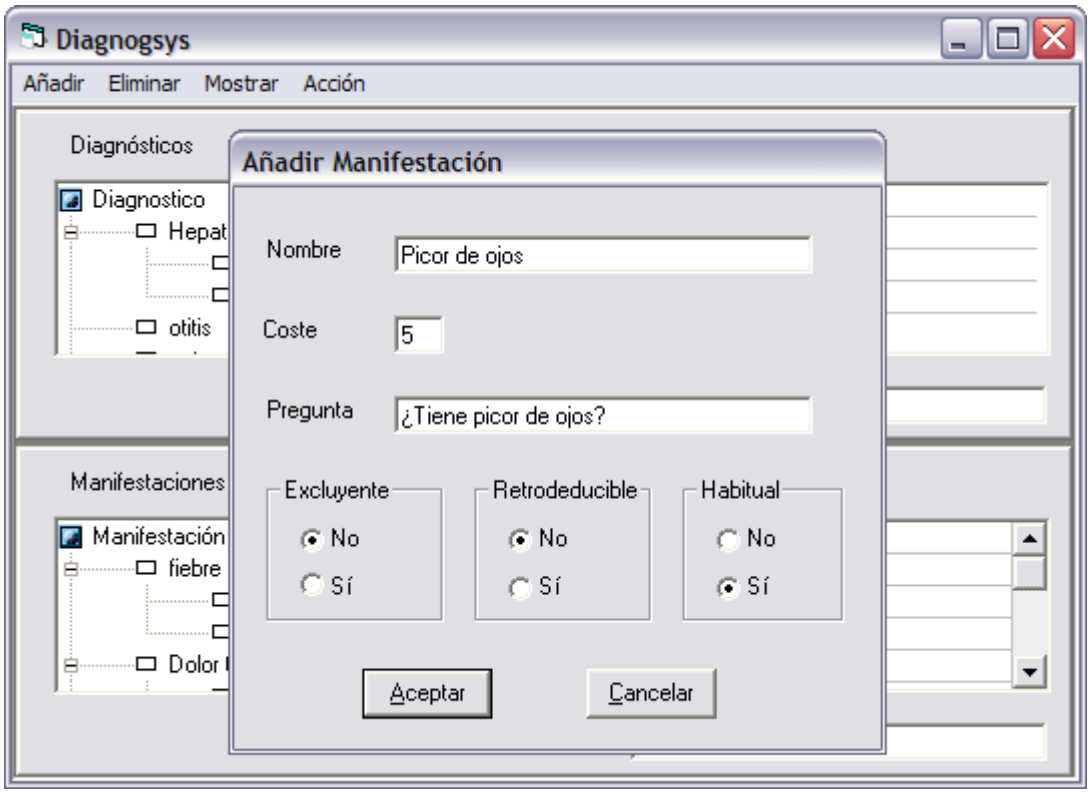


Figura 6.12. Ventana de solicitud de las propiedades de la nueva manifestación.

Una vez que el usuario pulse el botón Aceptar, la nueva manifestación se almacenará en la base de datos y se mostrará en el árbol de manifestaciones. En las propiedades, concretamente en la propiedad Hija de, podremos observar cómo aparece el nombre de la manifestación padre, si es que ésta no es la raíz, en cuyo caso esta propiedad aparecerá vacía.

6.4.2. Eliminar manifestación

Para eliminar una manifestación el usuario deberá seleccionarla y pulsar la opción del menú principal Eliminar → Manifestación.

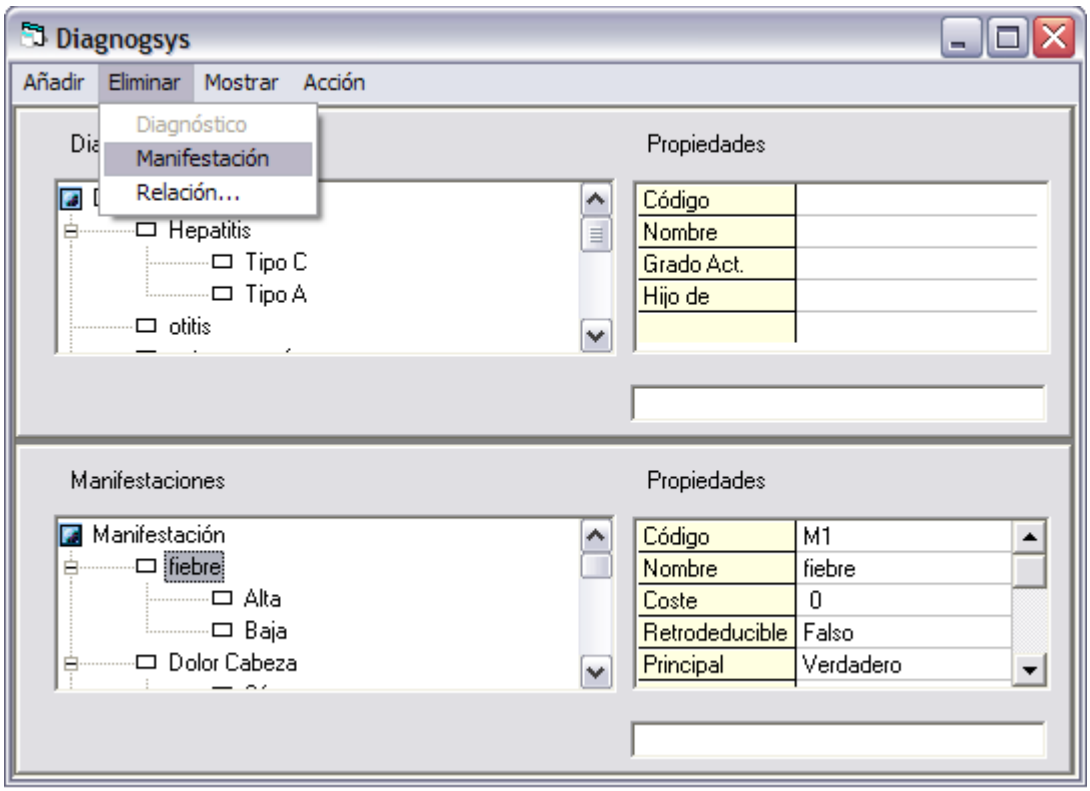


Figura 6.13. Opción de menú para eliminar la manifestación seleccionada.

De la misma manera que ocurre al eliminar un diagnóstico, el usuario debe tener en cuenta que al eliminar una manifestación se eliminarán todas aquellas manifestaciones que dependan de ella, es decir, se eliminarán todas las manifestaciones hijas de la manifestación que se va a eliminar, así como las relaciones que pudieran tener con los diagnósticos. Por este motivo aparecerá una ventana que advertirá al usuario de esta situación y le pedirá confirmación.

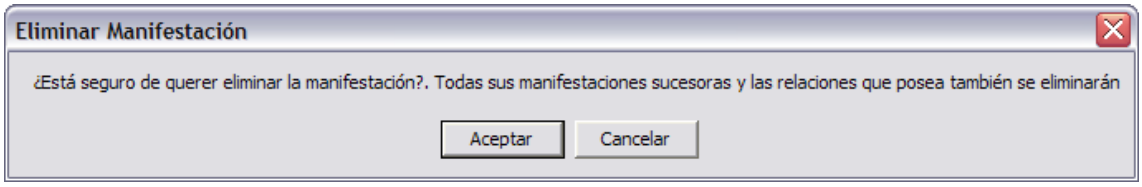


Figura 6.14. Ventana de advertencia de eliminación de manifestación.

Una vez eliminada la manifestación, esta y todas sus hijas, si las tuviera, desaparecerán del árbol de manifestaciones y de la base de datos, así como las relaciones que pudieran tener.

6.4.3. Modificar manifestación

Para modificar alguna de las propiedades de una manifestación deberá seleccionarse en el árbol de manifestaciones y pinchar la propiedad que deseemos modificar en el cuadro de propiedades. A continuación se deberá introducir el nuevo valor en el recuadro habilitado para esta acción y pulsar la tecla ENTER.

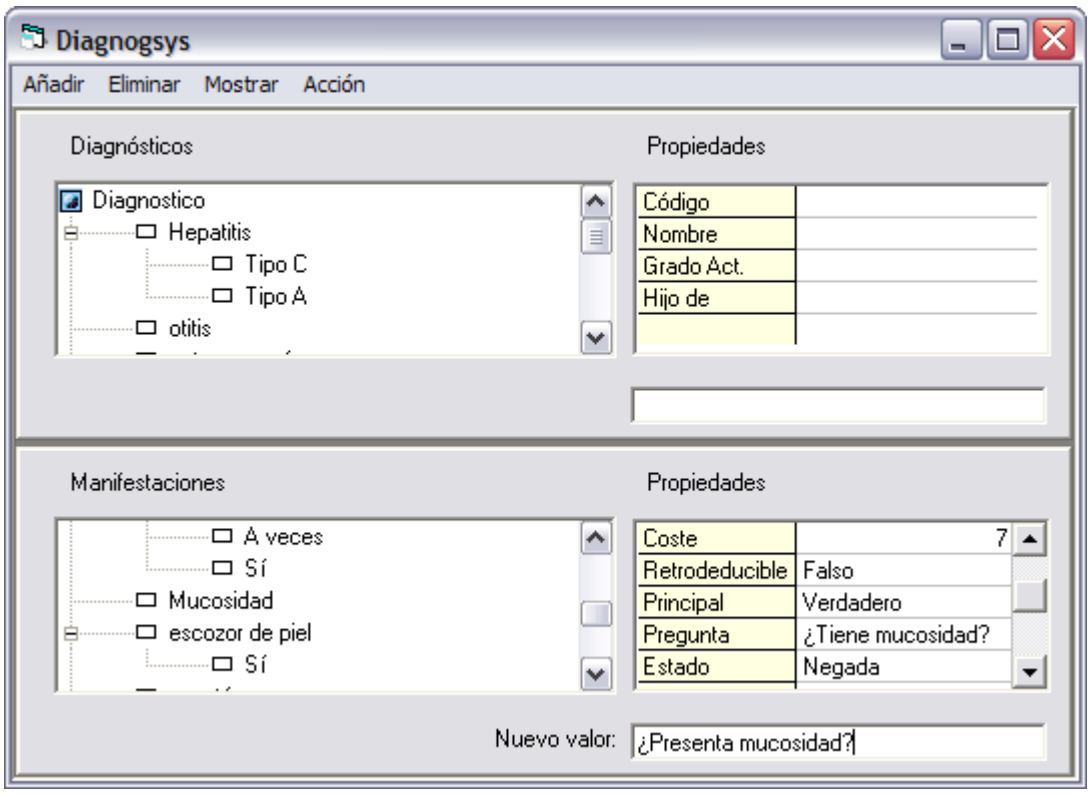


Figura 6.15. Modificando la propiedad pregunta de la manifestación Mucosidad.

6.4.4. Otras operaciones con manifestaciones

Las manifestaciones se pueden negar, activar o desactivar para reducir o aumentar la base inicial de hechos de la que va a partir el resolvedor de problemas. Así, si se desactiva una manifestación, ésta no va a estar presente en el problema a resolver. Si se activa, estará presente y activada, por lo que el resolvedor de problemas ya no preguntará por ella. Es equivalente a que se hubiera respondido afirmativamente a la pregunta que el resolvedor hubiera hecho al usuario para conocer si está o no presente en la base de hechos. Si se niega, se preguntará por ella durante el proceso de resolución.

Para activar, negar o desactivar una determinada manifestación, bastará con seleccionarla y pulsar el botón derecho del ratón, seleccionando la opción deseada.

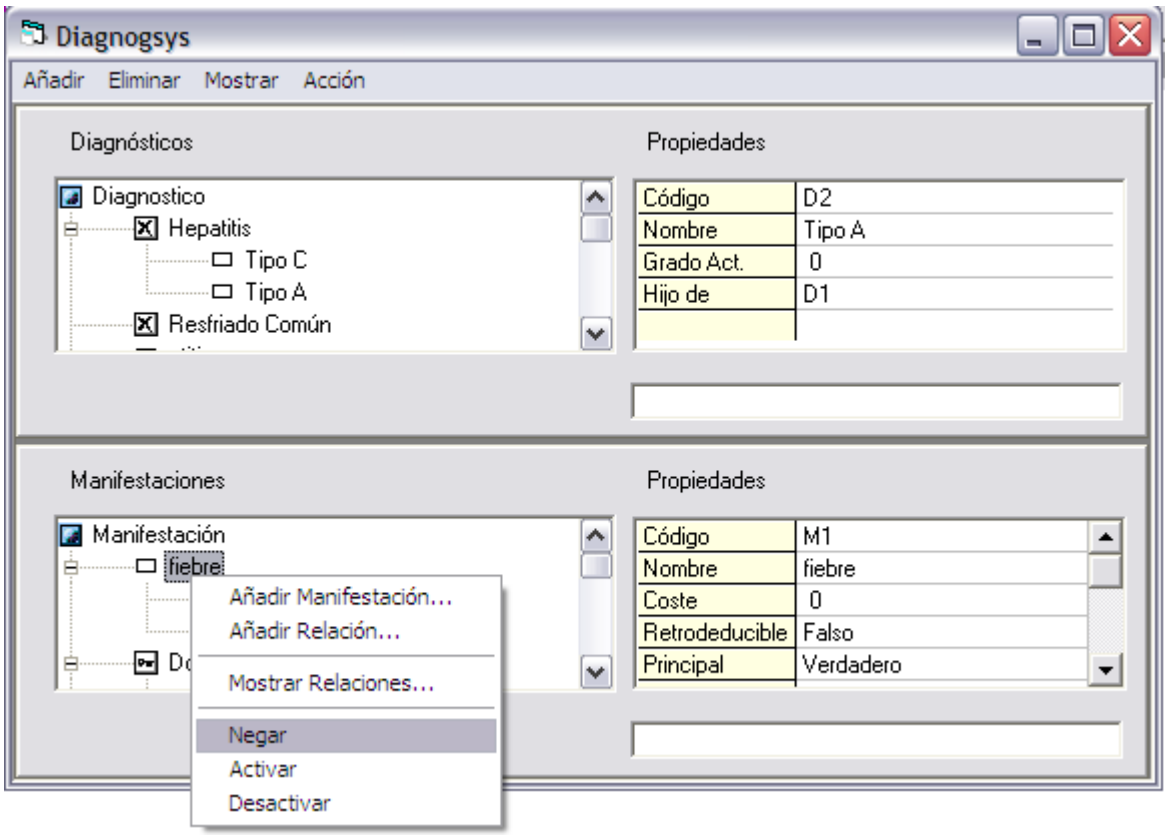


Figura 6.16. Opciones de negación, activación o desactivación de la manifestación Fiebre.

6.5. Relaciones

Los diagnósticos y las manifestaciones guardan cierta relación unos con otros. Así, una jaqueca estará relacionada con un dolor de cabeza.

Este tipo de relación entre diagnósticos y manifestaciones se establece mediante las relaciones.

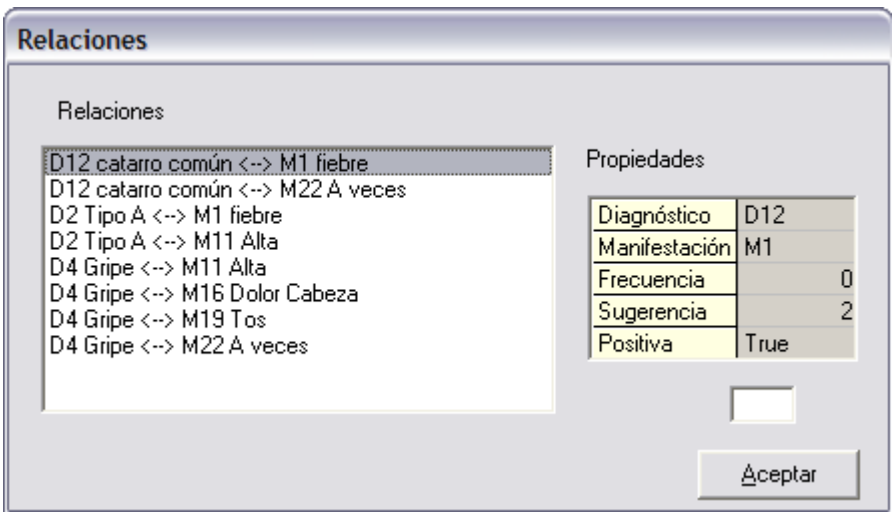


Figura 6.17. Ventana que muestra todas las relaciones.

6.5.1. Añadir relación

Para añadir una relación se puede optar por dos alternativas. Utilizar la opción del menú principal Añadir → Relación, o bien pulsar el botón derecho del ratón situando el puntero en cualquier lugar de uno de los dos árboles y eligiendo la opción Añadir Relación.

Al elegir la opción de añadir relación aparecerá una pantalla como la que se muestra en la Figura 6.18.

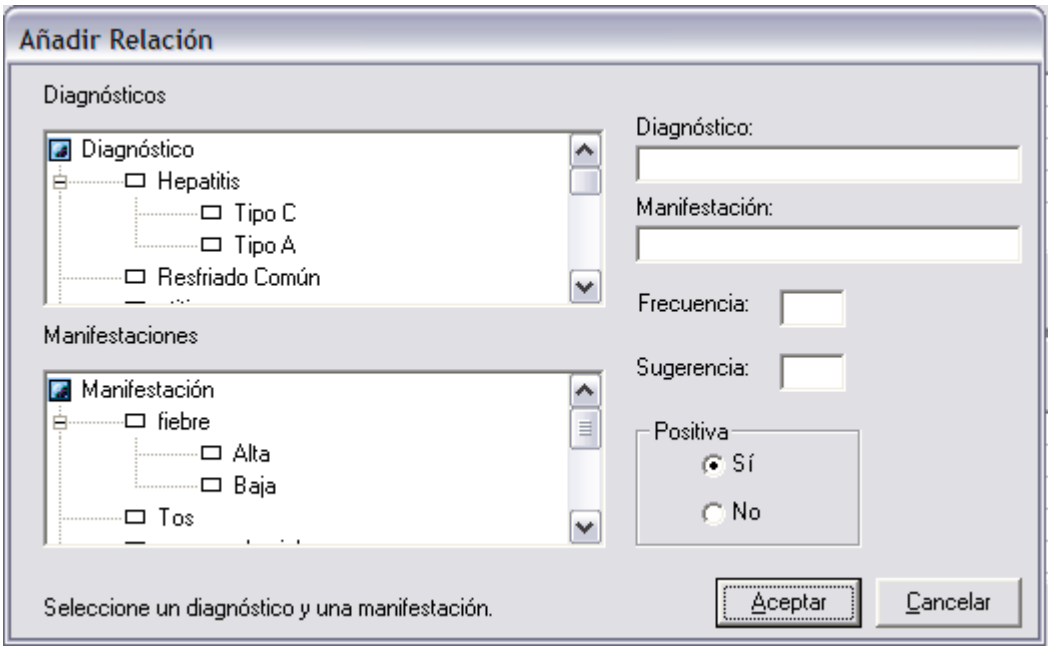


Figura 6.18. Ventana que permite al usuario añadir una relación.

En esta ventana el usuario deberá seleccionar un diagnóstico de los presentes en el árbol de diagnósticos y una manifestación de las del árbol de manifestaciones, así como introducir los valores de las propiedades de la relación.

Una vez el usuario pulse el botón aceptar la relación se creará y se almacenará en la base de datos.

6.5.2. Mostrar relaciones

Para poder visualizar las relaciones de un determinado diagnóstico o manifestación se deberá seleccionar el diagnóstico o manifestación deseada. Una vez seleccionada o seleccionado, la relación se podrá visualizar con la opción del menú principal Mostrar → Relaciones o bien situando el puntero del ratón sobre el elemento seleccionado y pulsando el botón derecho del ratón y seleccionando la opción Mostrar Relaciones.

Aparecerá una pantalla mostrando las relaciones tal y como se puede apreciar en la Figura 6.19.

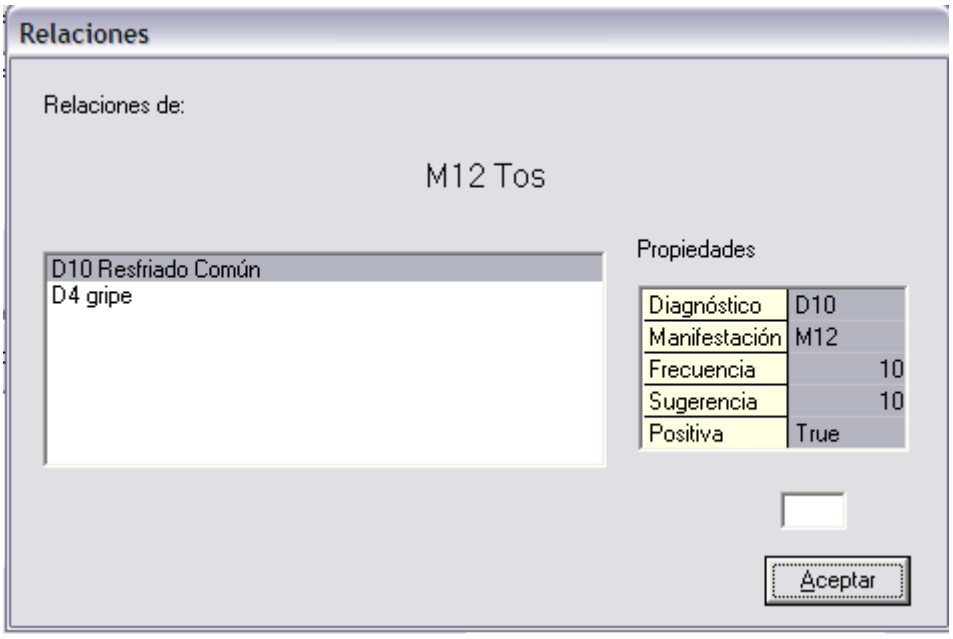


Figura 6.19. Relaciones de la manifestación Tos.

Si la manifestación seleccionada no tuviera relaciones se mostraría un texto indicando esta situación.

Desde esta pantalla se podrán modificar las propiedades de las relaciones de la manifestación seleccionada.

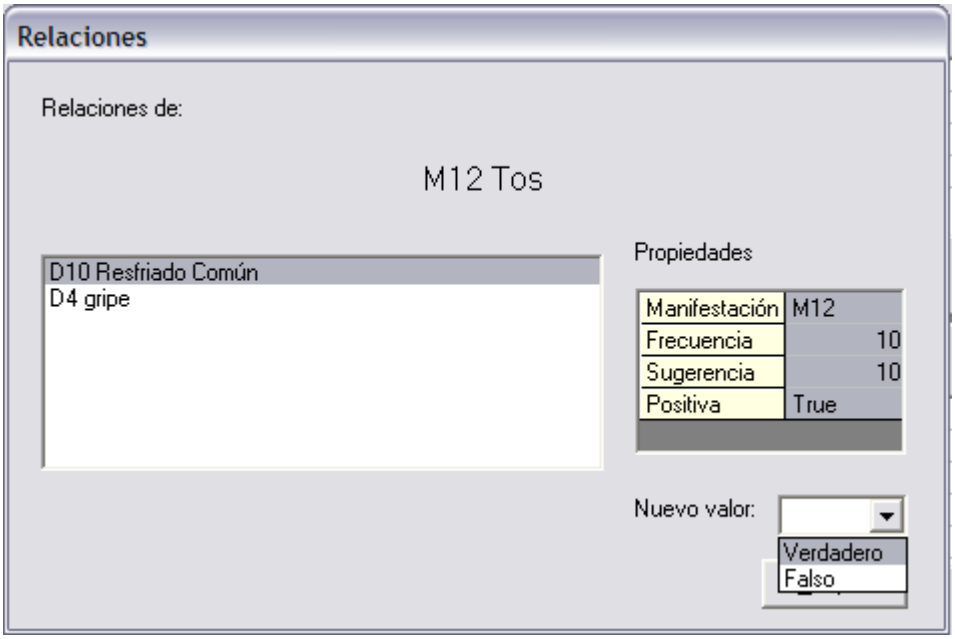


Figura 6.20. Nuevo valor para la propiedad Positiva de la relación de la manifestación Tos con el diagnóstico Resfriado Común.

También existe la posibilidad de visualizar todas las relaciones existentes entre los diagnósticos y las manifestaciones de ambos árboles pulsando sobre la opción del menú principal Mostrar → Todas las Relaciones.

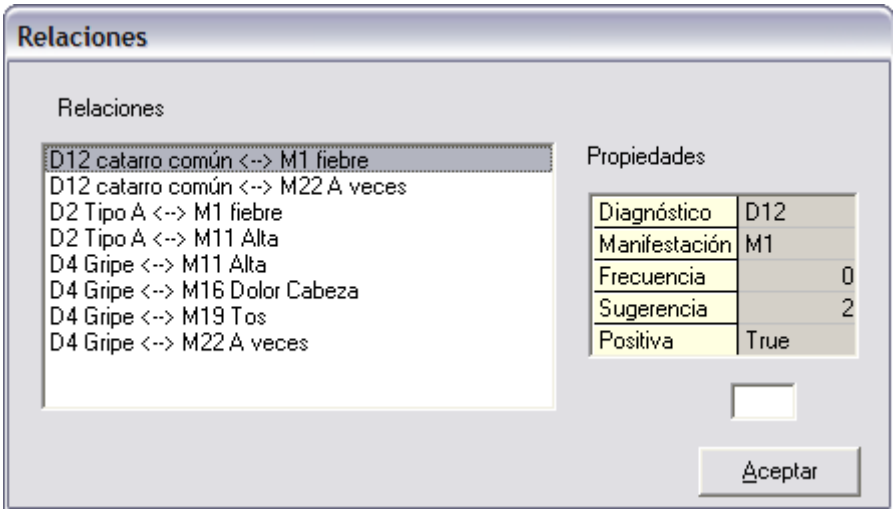


Figura 6.21. Ventana que muestra todas las relaciones.

6.6. Iniciar Resolvedor

Mediante esta opción del menú principal se iniciaría el proceso de resolución del problema y se ofrecería al usuario un diagnóstico de entre todos los posibles, acorde a las manifestaciones activadas.

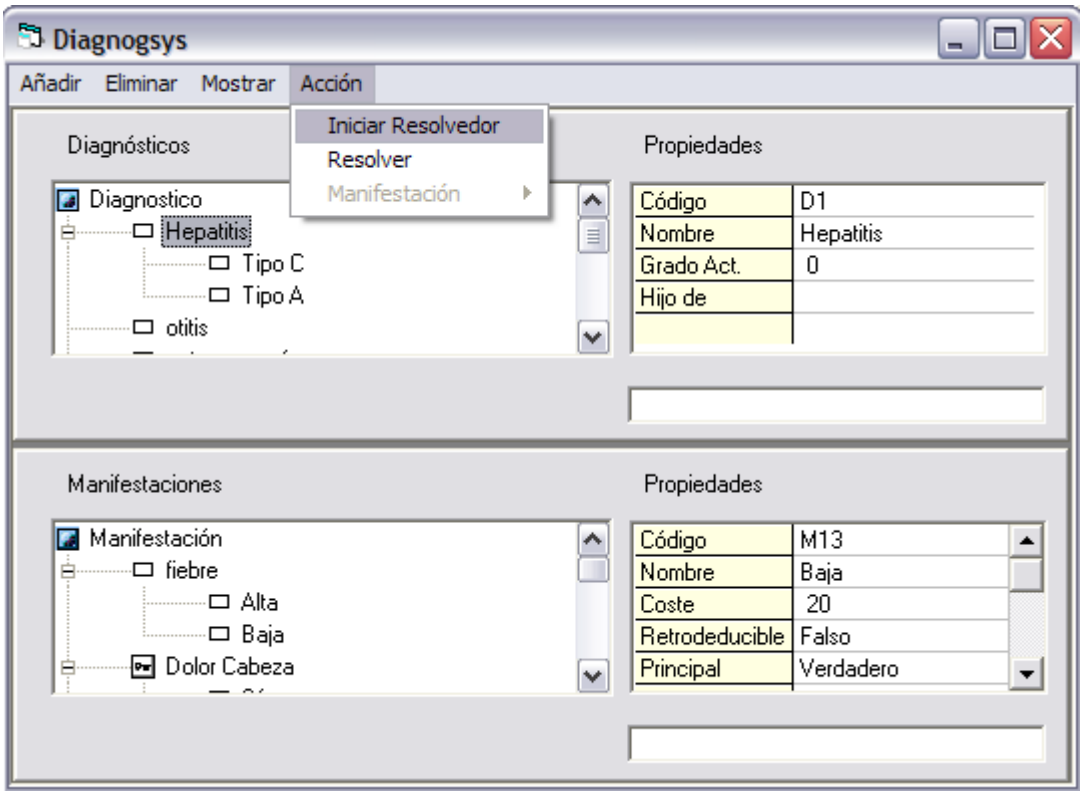


Figura 6.22. Se inicia el proceso de resolución del problema.

6.7. Detener Resolvedor

Durante la ejecución del proceso de resolución el usuario puede detenerlo en las ventanas que van apareciendo cuando se pregunta al usuario por la existencia o no de una determinada incidencia de la base de hechos.

Bastará con pulsar el botón “Parar Resolvedor” para detener el proceso.

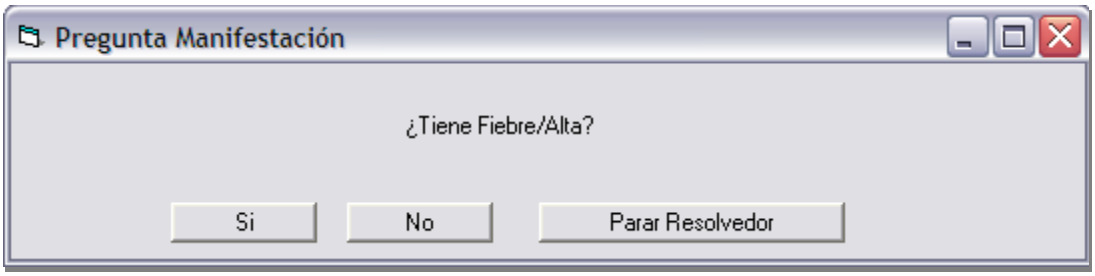


Figura 6.23. Botón “Parar Resolvedor”.

6.8. Resolver

Una vez detenido el proceso de resolución el usuario puede forzar al sistema a mostrar una solución y, por tanto, un diagnóstico.

Para ello, deberá seleccionar en el menú principal, la opción Resolver.

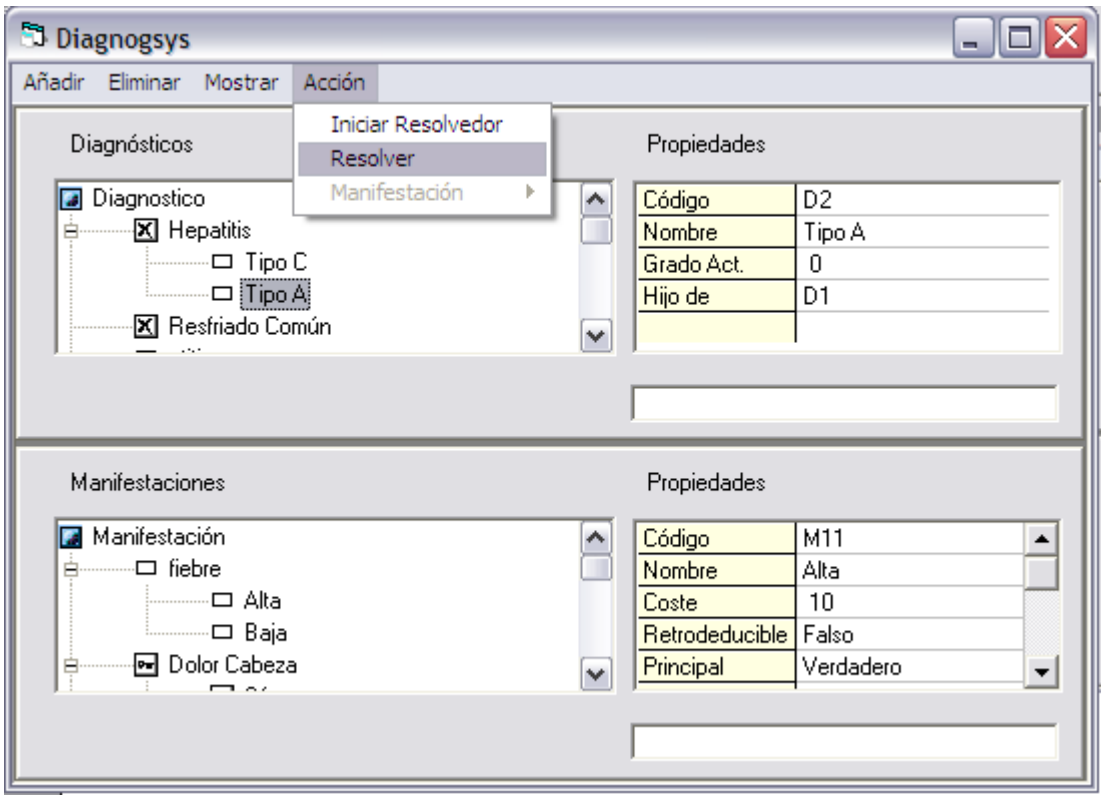


Figura 6.24. Opción Resolver.

7. CONSIDERACIONES FINALES

7.1. Conclusiones

Durante todo el proceso de análisis y de estudio, que ha sido necesario realizar para llevar a cabo el desarrollo del sistema, se ha llegado a varias conclusiones sobre las diferentes etapas del trabajo, de entre las que cabe destacar:

Dentro de la inteligencia artificial, los sistemas expertos están quedándose en un segundo plano con relación al rápido crecimiento en campos tan novedosos como el razonamiento subsimbólico o la visión artificial, que han surgido como nuevos campos de investigación dentro de este área.

Referente a lo leído y estudiado en relación con los sistemas expertos, se ha podido constatar que escasea la información relativa a sistemas basados en modelos, y que la mayoría de los libros tratan casi exclusivamente de sistemas basados en reglas, que son los sistemas más ampliamente conocidos y que poseen un mayor número de desarrollos.

Se ha visto también que los sistemas expertos son una buena manera de ayudar al experto en su trabajo, no de suplantarlo como mucha gente piensa, ya que estos sistemas no podrían existir sin la ayuda de los expertos. Además, un sistema experto nunca podrá aportar a una persona cualquiera, el conocimiento necesario, como para que se pueda prescindir del experto, ya que todo el proceso de razonamiento que se lleva a cabo desde que se plantea un problema, hasta la obtención de la solución, es transparente al usuario y, por ello, dicha persona no será capaz de resolver problemas parecidos a los que ya conoce, sino únicamente, los problemas aprendidos con el manejo diario del sistema, que es capaz de resolver fácilmente por la fuerza de la costumbre, no por un razonamiento paso a paso del problema.

En la cuestión sobre qué tipo de herramientas utilizar para el desarrollo del sistema experto Diagnogsys, se ha llegado a la conclusión de que ha sido mucho más complejo, pero, a la vez, mucho más interesante, utilizar un lenguaje de programación que ha supuesto desarrollar el resolvedor de problemas y la base de conocimientos desde cero, en lugar de utilizar herramientas que contenían un resolvedor de problemas y que están actualmente en el mercado. Además ha sido necesario diseñar y desarrollar la interfaz de usuario, por lo que el desarrollo del sistema Diagnogsys ha resultado ser un proyecto de gran envergadura, tanto desde el punto de vista del análisis previo, como del diseño, así como del desarrollo posterior.

7.2. Resultados

Una vez llevado a cabo el desarrollo y la implementación del sistema experto, los resultados a los que se han llegado después de probarlo son:

• El sistema desarrollado hace posible que la figura de una persona encargada de comunicarnos un posible diagnóstico acorde con las manifestaciones que comunicamos, pueda suplirse. Si nos centramos en el caso concreto que trata el proyecto, un servicio telefónico en el que una persona nos comunica qué enfermedad tenemos y el tratamiento que debemos realizar, según los síntomas o manifestaciones que le proporcionamos (servicio telefónico que actualmente ofrecen algunos seguros privados de salud), podría suplirse, en cierta medida, con el sistema experto Diagnogsys que se ha implementado.

• El sistema experto ha sido probado en diferentes ordenadores, tanto PCs como portátiles, con diferentes configuraciones, y se ha obtenido un funcionamiento óptimo. Se ha podido constatar que el sistema funciona correctamente sin necesidad de características técnicas del equipo demasiado exigentes. Más bien, al contrario.

• Por otro lado, se ha podido comprobar que la utilización del sistema es muy fácil e intuitiva, ya que cualquier persona con una lectura rápida del manual y proponiéndole una serie de manifestaciones iniciales, ha sido capaz de llevar a cabo una sesión con el sistema, sin necesidad de la ayuda de la persona que lo desarrolló.

7.3. Desarrollos Futuros

Dentro de los desarrollos futuros que se pueden llevar a cabo una vez finalizado el sistema se pueden proponer los siguientes:

• Mejorar la interfaz de usuario, tanto en el apartado visual como en el apartado funcional, dotando a los menús emergentes de mayores funciones recogidas en el menú principal, por ejemplo.

• Incorporar una ayuda del sistema que guíe al usuario tanto a la hora de confeccionar e introducir en el sistema la base de hechos como durante el proceso de resolución del problema.

• Mejorar la actualización de la Base de Datos. Los datos se cargan y se actualizan sobre una base de datos Access sin posibilidad de cambiarle el nombre ni la ruta en la que se encuentra el fichero Access (el fichero debe encontrarse en el mismo directorio que el archivo ejecutable de la aplicación) y sin posibilidad de tener varios ficheros con distintas bases de hechos.

8. BIBLIOGRAFÍA

- Adarraga, P. y Zaccagnini, J.L. (1994). Psicología e Inteligencia Artificial. Madrid: Trotta S.A.
- Batnov, D., Nagarur, N. and Nitikhunkasem, P. (1993). EXPERT-M: A knowledge-based system for maintenance managenet. Artificial Intelligence in Engineering. 8 (1993)
- Castillo, E. y Alvarez, E. (1989). Sistemas expertos aprendizaje e incertidumbre. Madrid: Paraninfo S.A.
- Consejo Superior de investigaciones científicas. (1989). Inteligencia Artificial para la gestión de bases de datos. Madrid: Fundación FUINCA
- Cuenca, J. y otros. (1987). Inteligencia Artificial, Sistemas Expertos. Madrid: Alianza-Informática S.A.
- Gevarter, W. M. (1987). Máquinas Inteligentes: una panorámica de la inteligencia artificial y de la robótica. Madrid: Díaz de Santos S.A.
- Maté Hernández, J.L. y Pazos Sierra, J. (1988). Ingeniería del conocimiento: diseño y construcción de sistemas expertos. Córdoba, Argentina: Sociedad para estudios pedagógicos
- SAMPER, J. J. "Introducción a los sistemas expertos". 2009

  (http://www.redcientifica.com/doc/doc199908210001.html)
- ALONSO, D. "Sistemas expertos". 2001

  (http://es.geocities.com/denisalonso2001/SISTEMAS_EXPERTOS.htm)
- Wikipedia (http://es.wikipedia.org/wiki/Sistema_experto).
- Wikidoc (http://wikidoc.org/index.php/Main_Page).
- Tutorial de Lógica Difusa por el ingeniero Oscar G. Duarte

  www.geocities.com/programa_educar/tutorial3.htm
- Grupo Iberoamericano de Aplicaciones con Lógica Difusa

  www.imse.cnm.es/~gial/

ANEXO 1: PRESUPUESTO ESTIMADO DEL PROYECTO

Para realizar una buena estimación de los costes, se han estimado, por un lado, los costes generados por los recursos humanos y, por otro lado, los costes que provienen de los recursos materiales.

A todos y cada uno de los costes incluidos en este apartado se les ha añadido el impuesto sobre el valor añadido (IVA).

En primer lugar nos centraremos en los costes procedentes de los recursos humanos y para ello haremos uso de la herramienta MS Proyect 2010. Tras haber finalizado la planificación del proyecto procedemos a asignar los recursos humanos a cada una de las tareas indicando el tiempo dedicado de cada persona a cada una de ellas y los resultados obtenidos son los que se muestran en la tabla 1.

Personal	Horas trabajadas	Precio/hora	Total
Director del proyecto	61,5 h	35 €/h	2.152,50 €
Jefe de proyecto	283,5 h	27 €/h	7.654,50 €
Jefe de calidad	182 h	25 €/h	4.550,00 €
Diseñador	155 h	20 €/h	3.100,00 €
Analista	174 h	22 €/h	3.828,00 €
Programador	527 h	15 €/h	7.905,00 €
Encargado de Pruebas	317,5 h	16 €/h	5.080,00 €
Encargado del aseguramiento de la calidad	118 h	18 €/h	2.124,00 €
Total del proyecto	**1818,5 h**		**36.394,00 €**

Tabla 1: Estimación de costes de los recursos humanos

Como puede observarse en la tabla 1, el coste total procedente de los recursos humanos es de 36.394,00 €. En ella también puede verse el tiempo dedicado al proyecto de cada una de las personas, el salario que cobra por hora y el salario total por su participación en el proyecto.

Para estimar los costes procedentes de los recursos materiales tendremos en cuenta tanto el gasto producido por el software como por el hardware. En la tabla 2 puede verse el material necesario para la realización del proyecto, así como el coste de cada uno de los recursos.

Material	Coste
Licencia Windows Vista SP1	120,00 €
Licencia MS Proyect 2010	25,00 €
Licencia MS Office 2007	139,00 €
Licencia MS Visual Basic 6	150,00 €
Ordenador Personal	1.200,00 €
Material fungible	50,00 €
Total del proyecto	**1.624,00 €**

Tabla 2: Estimación de costes de los recursos materiales

Como puede verse en la tabla 2, los costes generados por los recursos materiales ascienden a 1.624,00 €. Por lo tanto, teniendo en cuenta el gasto generado tanto por los recursos materiales como por los recursos humanos, el coste total estimado para el proyecto es de 38.018,00 €.